

 ATARI
COMPUTER
ENTHUSIASTS

3662 Vine Maple Dr. Eugene OR 97405

APRIL, 1983

Mike Dunn & Jim Bumpas, Editors



**ATARI[®] COMPUTER
ENTHUSIASTS**

Official Users' Group Logo

NEWS AND REVIEWS

For the first time since the ACE Newsletter was started (almost 3 years!) I was not able to go to the San Francisco Computer Faire. I will leave the report to our President, Kirt Stockwell, who will give you all the latest info from Atari and the Faire. Recent events at Atari are not very encouraging; these comments are written before the Faire, so hopefully Kirt can find out the real answers and set the record straight. By now, I am sure you heard about the big layoff and move to Formosa of the Computer Division.

Because of this move, many Atari products, such as the 850 interface, are in very short supply and may remain so. The "new" 800 is rumored to be very different — no expansion slots, as there will be only one memory board, with the others not hooked up, no right ROM slot, and I hear no joystick ports 3 & 4. According to Atari, since they can produce the 800 cheaper overseas, they will not be changing the 800, but it will remain the same.

I also hear that much third party software won't run on the Atari 1200, because of the tendency of machine language programmers to use illegal entry points. With the new ones being made in the Orient, the price should come way down which leads to another problem. In our area, as well as many others, according to other newsletters, computer stores are dropping the Atari line because of the price. With Penney's, etc, selling 800's with 48K for \$499, which is only a few dollars over the dealer's cost, a computer dealer cannot afford to sell them. No one expects the Penney's salesman to spend time with them to teach them to use their computer, but they do expect the computer dealer to do so, and he can't at a \$20 or so markup. The user groups are the only place many new owners can turn to, but, of course, the owners are getting tremendous bargains — I paid \$1200 for my Atari 800 a little over 2 years ago. I hope Kirt can set us straight on all of this in his visit to Atari and the SF Faire.

ANTIC (600 18th St., San Francisco, CA 94107) is now monthly for \$24 a year. Started by the editor of a newsletter similar to ours in San Francisco, Jim Capparelli, it has become very successful but has kept his ideals — all the programs are in the public domain. If I was to make ACE a business (which will never happen!!), it would be like ANTIC. This month is their one-year birthday, so Happy Birthday and many more.

An excellent new guide to computer books has just been published for only \$2.00. Titled **Computers, a Comprehensive Guide**, it is available from the Yes! Bookshop, 1035 31st St. NW, Washington, DC 20007. You can order the books from them, but the guide is very good — I have many of the books reviewed, and agree with the reviews and the ratings they have to indicate the best ones on a subject. Another book I received for review several months ago was **The Computer Coloring Book** Prentice Hall, \$6.95, 60 pages. I couldn't figure out who to get to review it. Kids couldn't understand it and adults didn't even want to look at it. So I stalled, and got another letter from the publisher asking how come we had not reviewed it yet. So here it is. A large, nicely done book, with the alphabet and a picture to color. It has a glossary which is "The computer glossary for everyone... effectively demystifies the jargon... that children should start on the road to computer literacy as soon as possible... a needed teaching device for children... etc" (all from their press release. Each letter has a word, definition and a picture to color. As an example, "Q" has Query. "Query is a request for information. Query programs let you view the information that is On-Line in your Computer System. A Query Language, designed as part of a Query Program, lets you request your information in many different ways..." etc. The picture to color shows a picture of a computer in a chair getting the fourth degree from some policemen with billy clubs.

More on the **ATR 8000** (Software Publishers, 2500 Randol Mill Rd., Suite 125, Arlington, TX 76011): last month we began with a short summary of this new product, a combination disk drive interface, printer and RS 232 interface, 48K printer spooler and CP/M adapter. Since I bought one of the first ones out, there were a few bugs, which have now been corrected. A new ROM has solved the problem of some Atari disks not loading, and some of the other odd problems in the Atari mode. In the CP/M mode, we have run Osborne and KayPro CP/M programs, and I understand Xerox programs will run also. They are about to come out with a version with only the interface, without CP/M, at a very attractive price. This will be upgradable to the more expensive model by the owner. New programs almost ready include a Modem program, disk emulator programs for many different CP/M computers, and others. There is also a new board, called Co-power 88 allowing you to run 8088 IBM PC programs with up to 256K and MS-DOS!!

Using CP/M with 40 columns is not ideal, but **Austin Franklin** (43 Grove St., Ayer, MA 01432) is about to release the **Austin 80**, an 80 column board using an RGB and emulating the DEC 101 terminal! This means you can run the 80 columns on a RGB color monitor at over 600 lines an inch resolution — much clearer than any color TV can do. Only \$289 retail — a review as soon as we get one.

Speaking of CP/M, I can't imagine how a novice could get a KayPro or Osborne computer and get it to work. If you have ever seen the Atari Word Processor, with all its commands and its huge instruction manuals, etc. — that would be a simple, easy to use CP/M program!! Great for programmers — you can do anything, if you know machine language.

Several issues back we reported on a new OS board allowing you to use custom OS EPROMS, etc from Newell Industries (3340 Nottingham, Plano TX 75074) which included the **Ommimon** monitor from David Young. This chip uses the hidden 4K of the OS, and allows you to use many machine language functions on programs which are difficult otherwise. Since it uses the "hidden" memory, it can be called any time, with any program. It is now available without the OS board and can plug into either a 400 or 800, for \$99 retail. Available from Newell Industries and others. Also from Texas is a new, inexpensive **EPROM Burner** from Creative Firmware (707 Auburn Dr., Richardson, TX 75081) for \$80 with an adapter for 2732's an additional \$17.50. They also sell Cartridge Boards, EPROM's, etc.

This article was written on the new **Bank Street Writer**, Broderbund Software, 1938 Fourth St., San Rafael, CA 94901, \$69. This is a new word-processor, written for children, and reviewed in TIME, etc., and picked by Atari to use in their childrens Computer Camps this summer. We just received it today, and will review it next month after we use it — probably by a person in the age group it was written for.

—Mike Dunn

BRYAN'S ARCADE

This month I received two new games from different companies. The games are a brand new **Jawbreaker** from Sierra On-Line and **Astro Chase** from First Star Software. I also had to go out and buy a copy of **Miner 2049'er**. I spent my \$50 very well.

MINER 2049'er

If you are a Donkey Kong fan and you don't want to wait for Atari to release their version of D.K. then **Miner 2049'er** is definitely what you are looking for. In my opinion this is the best climbing and jumping game out for the Atari 400/800 yet. This game is awesome!

In the game you are **Bounty Bob** in a futuristic mine shaft trying to get out! You have to avoid radioactive waste, deadly radioactive mutants and you have to avoid falling from high places. The game has 10 different levels or what they call "stations" and 10 different zones. So if you get past station 10 zone #1 you then go to station 1 zone #2. So far I have gotten to zone 4 station 8 with a score well over 200,000.

The object in the game is not to get to the top of the screen. The object is to go over every little step of the screen to make all the girders a solid color instead of the polka dot color they are to begin with. Instead of the hammer like **Donkey Kong** you have to pick up various tools along the way. When you pick up one of the tools the radioactive mutants turn green and have a smile face. When they are that way you can kill them by running into them!

Over all I think **Miner 2049'er** is one of the best if not the best game for the Atari home computer. The graphics in the game are very good; the sound is also very good. On a scale of 1 to 10 I give **Miner 2049'er** a 9.5! Yes it's that good! It's definitely worth the money.

Miner 2049'er is available from **Big Five Software** on a ROM cartridge for \$49.95.

I also received **Astro Chase** from **First Star Software**. **Astro Chase** is programmed by **Fernando Herrera**. **Fernando Herrera** was the first person to win the Atari Star award for his outstanding program **My First Alphabet**.

Astro Chase puts you in charge of a space ship defending our home planet, Earth, from **Megamines** which are programmed to destroy Earth. The megamines head straight towards Earth and one megamine alone has enough power to destroy Earth. Your object is to destroy the megamines before they destroy Earth. There are 16 megamines per level and you have to destroy all 16 of them before you advance to the next level.

At the start of the game you get 3 ships with 1000 units of fuel per ship, each time you fire you lose one unit of fuel so you tend to lose fuel rapidly. In each of the four corners of the galaxy there is a fuel station. Each time you run over the station you get some more fuel and you can keep restocking fuel until you feel you have enough fuel to go on. During the game other things besides megamines attack you. Alien spaceships try to ram into or shoot and kill you. You can either shoot them or you can run over a shield which enables you to make contact with them without getting killed. The shield also protects you from their bullets which is also very helpful.

Astro Chase has outstanding graphics. After every 4 levels there is an intermission which shows you coming back to Earth and you do a bunch of weird things. The intermissions are very entertaining and give you a chance to rest your hand after continuous play. There are 7 intermissions in all and they get better every time from beginning to end. The sound in **Astro Chase** is also pretty good. On a scale of 1 to 10 I give **Astro Chase** an 8.5!

I also got a review copy of the all new **Jawbreaker** from **Sierra On-Line**.

Jawbreaker is another **Pac-Man** type game with a catch. In this game you are about 3 or 4 times as big as the normal sized **Jawbreaker** or **Pac-Man**. The object of the game is to clear the screen of all the little dots, which are supposed to be jawbreakers, without getting killed by the round monsters. You can kill the monsters temporarily by eating one of the swirling lifesavers. The lifesavers makes them turn red and then you can eat them. If you are good enough to clear an entire screen of jawbreakers and lifesavers then there is a little intermission. In the intermission a little tooth brush comes out and brushes your teeth for you. But if you happen to get killed by a monster then all of your teeth fall out.

Overall **Jawbreaker** is a very "cute" game your whole family should enjoy. The graphics aren't outstanding but they aren't bad. The sound is very good. While you are playing a little tune goes on in the background. On a scale of 1 to 10 I rate **Jawbreaker** an 8.

Next month I hope to review **Zaxxon** the popular arcade game now out for the Atari from **Data-Soft Inc.**, and any other games I should happen to receive.

Rumors... Atari is supposedly going to release **Dig-Dug**, **Donkey Kong**, and **Kangaroo** in the next month or two. Look for them soon! **Gebelli Software** is supposed to release a **Donkey Kong** type game called **Candy Factory** look for that soon too!

That's all for this month!

—Bryan Dunn, Games Editor

WEST COAST COMPUTER FAIRE

Kirt E. Stockwell

It's hard to know where to start, so I'll start with our first organized activity after we all arrived in San Francisco. Following our founder and first President, STACY GOFF driving his Great Racing Rabbit, we proceeded to get lost on our way to ATARI. After about 20 minutes of zany driving, we finally found ATARI and went in. We were met by Earl Rice and Mark Cater. They showed us the new VIDEO VISIT tapes, the 1200XL, the new logo, and the CORVUS hard disk the ATARI Bulletin Board System uses. We spent some time talking to Gary Furr, designer of the new ATARIWRITER word processing program. The new Director of Marketing spent some time with us and answered such questions as he safely could.

Many of you may have noticed the 850 interface can be hard to obtain. At least it is here in Oregon. The rumor ATARI is halting production on the 850 was denied, at least for the present. The official position is the 850 will continue in production until such time as ATARI feels there is an adequate installed base of 1200XL's. What criteria will be used to determine "adequate installed base" was not specified. Personally, I don't expect to see the 850 in production too much longer, as there are too many other peripherals on the market which eliminate the need for the 850. These include the ATR 8000 and the MICROBITS MPP 1000 printer interface, among others.

The word at ATARI is the 800 will continue to have slots for extra boards, as well as having the right-hand cartridge slot connected. The lower cost of production made possible by the shift to Singapore (or wherever they went) should provide ATARI with the capability of reaping a profit without cutting the guts out of the equipment. Officially, ATARI has no plan or desire to replace the 800 with anything. (Unofficially, I am persuaded ATARI is working on at least one application-oriented computer. This powerful new machine will probably be aimed at the HOME business market.)

The new VIDEO VISIT tapes are a gas. Chris Crawford is one of the most pleasantly insane people I have ever met. As a tool for raising interest levels and teaching broad concepts, the tapes are quite nice, as well as fun. Also on the tapes is the NEW IMPROVED ATARI APPROVED ACE logo. This new logo, which you might have seen by now in the ATARI CONNECTION magazine, is animated. One of you hot-dog programmers out there might want to write a program to produce the animated logo on the screen (for the MENU cover, for instance) and send it in to us. The plan is eventually all ACE groups will use the approved logo on their letterhead, newsletters, and such. We'll see.

I might mention, while we were in the Bay Area, we were able to look at the insides of an ATARI 1200XL. There is only one board, with remarkably few chips, and only 2 connectors from the top cover which holds the keyboard. The entire thing is shielded to minimize RF garbage, and the power board inside is mounted on a herky cooling fin. Contrary to some of the rumors going around, the 1200XL DOES support RGB monitors through the monitor port. Several programs written by third parties, including DATA-PERFECT and LETTER-PERFECT won't run on the 1200XL. Neither will METEOR STORM written by Jon Attack and L.J. Knoll. The problem here is the authors used illegal entry points to operating-system routines, and the vectors on the 1200XL's operating system are different.

ON TO THE FAIR!!

The first stop, naturally, was the ATARI booth. ATARI was displaying new software, and always had large crowds around, but I saw nothing new in the way of hardware. From there, things become a blur. There were many new and exciting products on the market, both software and hardware. I'll try to list some of the more note-worthy. A new computer called "ACCESS" with 9" amber monitor, detachable keyboard with 10-key pad, an EPSON MX80 built into the top, both acoustic and direct-connect modems also built in, weighing about 25 pounds. The KAYPRO 10, basically the KAYPRO II but having only ONE floppy; the other was replaced with a built-in 10 MEG hard disk, still complete with the PERFECT-SOFTWARE package, all for under \$2600.00. Epson didn't show up as a corporate identity, however there were companies showing the new EPSON computer...cute. APPLE was displaying and demonstrating their new LISA. Both Stacy Goff and myself tried, at different times, to get close enough to take pictures or shoot video. No such luck. There was a core of APPLE fanatics around that machine the AYATOLLA KHOMEINI would have been proud of.

Top Honors for display quality and effectiveness should probably go to PERFECT SOFTWARE. Their area was in a room accessed by a long hallway. Lights were running along the floor on both sides of the hallway, much like a runway. There were monitors every few feet, with a well produced film constantly showing. Inside, a green laser fired random bursts of patterns over the room at a safe level. Big screen TV inside showed the same video the outside monitors were displaying. Well-informed software specialists from PERFECT SOFTWARE were demonstrating their wares and talking OEM packages with manufacturers.

There were too many new computers, software and peripheral products on the market to even mention. Some of them are well thought out, well made, and competitively priced. Some are candidates for non-existence in the near future. RADIO-SHACK, for instance, had very little of interest to ANYBODY at their booth. They don't seem to have kept abreast of developments in the field. APPLE, on the other hand, was displaying both the new LISA and the APPLE 2E. Several vendors were selling APPLE 2+ machines at ridiculously low prices; APPLE seems to be dumping the older units to make way for the new.

MICROBITS, friends of ours from ALBANY, OREGON, had a very small but effective booth. Their products, the MPP1000 Interface and the MPP1100 MODEM, don't take much room to show. Several manufacturers saw enough to order their interface unit, and many dealers also placed orders. So the lack of the 850 from ATARI shouldn't be a problem, as the MICROBITS equipment, which is quite good, will be available locally soon.

One of the high points of the FAIRE for me was my stop at the COMPUTE booth. I struck up a conversation with SCOTT CARD, book editor, concerning the market for application software. I soon discovered I was talking to ORSON SCOTT CARD, one of my favorite SCI-FI authors. Had a nice long chat and came away with many great program ideas. Now if only I had time to write the programs.

Also stopped at the DATASOFT booth. Saw ZAXXON running simultaneously on the VIC-20, Radio-Shack Color Computer, and ATARI 400. Guess which one looked the best????? The VIC doesn't seem to be able to handle fine scrolling. The Color-Computer is a bit slower than the ATARI, and the colors are more limited. Also heard DATASOFT had a copy of the 400-800 version of DEFENDER, but didn't get to see it.

Saturday afternoon ATARI held a reception for User Group Officers at the Golden Gateway Holiday Inn. Here they demonstrated ATARIWRITER and ATARI LOGO (the language). Hors d'oeuvres and drinks were served, and we all spent several fun hours swapping outrageous lies and rumors. I met, among other ACE presidents, Dr. Mel Boreham, pres. of the TRANSISTHMIAN ACE. (TRACE. They're based at the Panama Canal). Also Dave Mentley of ABACUS, and a pile of other presidents from around the country.

This being my first time out in the Bay Area, I was pleased at the nice things so many people said about our newsletter. Had several corporate groups, as well as a few advertising types ask about the availability of space in our newsletter. Told them thanks, but no thanks. (Guess I'll have to get rich some other way.)

Had a great time, walked my legs off, and learned quite a bit about the new direction in application programming for PERSONAL computers. Don't have time or room enough to talk about that now, will try to get to it next month. By for now...



MICROBITS PRINTER INTERFACE

I am pleased at this time to review what I feel to be an important new piece of equipment. For those of you who have yet to purchase a printer and interface, or who have purchased a printer and can't get an ATARI 850 interface, listen up!

You may have read my previous review on the MICROBITS MODEM, which plugs into the #4 joystick port, comes complete with excellent software, and doesn't need an interface. If so, you may be prepared for the new MICROBITS MPP1100 parallel printer interface.

This little jewel plugs into the #3 joystick port, needs no power supply, connects directly to your printer, and retails for only \$99.00. Among other sterling qualities, this unit is, according to MICROBITS, compatible with any parallel printer, and will function with ANY software. I have one on my 800, and am quite pleased with it for several reasons: 1. No power supply or batteries; 2. No cables looped around everywhere; 3. No switches to remember; 4. It is virtually invisible when the system is assembled; 5. The price is very reasonable; 6. They are available.

I don't know about the rest of you, but I feel galled every time I think about the marketing practices of the computer industry. ATARI is no worse than most, but no better. My technical sources tell me that any personal computer manufacturer could build printer interfaces and direct-connect modems into their units for \$15.00 or \$20.00 per unit. Instead, they choose to make costly peripheral devices, on the basis that each unit is another sale, and another opportunity to hit the customer for a hefty markup. But, since the manufacturers chose their own way, third party equipment builders must basically re-invent the wheel, and add-on peripherals can't be as inexpensive as built-ins would have been.

MICROBITS struck a great design compromise with the MPP1100. They have re-written one of the operating system ROM chips, which they provide the replacement for when you buy the interface. The new chip, when installed in the operating system board, functions in all other respects identically to the old chip. Only the printer handler routines were changed. The ATARI, thus, does the majority of the work for the interface. As a result, the interface unit is only about 1.5 X 2.5 X 3.5 inches, with a short cable from the box to the printer, and a long cable from the box to the computer. The flat cable runs under the console nicely, disappearing along with the box, which tucks right behind my EPSON MX80.

I have used my MPP1100 with all the different word processors I could get my hands on, and there are no problems. According to MICROBITS, not only will the interface work on ANY software, but it will transfer data faster than the ATARI 850. For those of you with buffered printers, the buffers load faster, giving you more time with the CPU.

Some of you out there might need all those extra ports the 850 offers. I suppose you could run your train set with them or something. But my personal choice is the MPP1100.

—Kirt E. Stockwell

ANATOMY OF DOGGIES

(by Garry Francis; reprinted from ACE N.S.W., Australia)

Who the Hell is Stan Ockers Anyway?

Way back in June 1981, the Eugene ACE Newsletter published a lunar lander program written by a fellow named Stan Ockers. This in itself was not unusual, but Stan returned with another program in the next issue. And the next. And the next. In fact, this prolific programmer from Lockport, IL has now written a program (or sometimes two) for every single issue of the ACE Newsletter since June 1981.

In December 1981, editor Mike Dunn awarded Stan with the First Editor's Award. His work is often reprinted in ANTIC and the Michigan ACE Newsletter and some programs (such as Chicken) are already considered classics.

Stan's programs may be divided into 2 very distinct categories — games and utilities. They are like a developing series of tutorials in advanced graphics techniques from BASIC. His philosophy has been "...perhaps the best way to explain what is going on is to go through an actual program..." But even if you don't want to learn about fancy graphics, the programs are still fun to use.

I have always admired Stan Ockers' work, but I was astounded to find out nearly all his programs were developed on a cassette based Atari 800 with 16k RAM and a black and white TV! He didn't even have an assembler! The moral is, of course, you don't need 256k and a quadruple density disc drive to write good programs, as many people seem to think. You merely need a good imagination and a lot of hard work.

Enter Doggies

Doggies has always been one of my favourite of Stan Ockers' games. I originally looked through the code to see how on earth it worked. Just recently, I decided it is an excellent program for the magazine. Stan has often pointed out minor deficiencies (such as lack of color), which he leaves for readers to fix themselves. So I set about to do a bit of polishing up to Doggies. Unfortunately, my polishing up got out of hand and I have now completely restructured the program, removed a couple of minor bugs, improved the response time, screen display and colors and renumbered it. But despite the weeks of work I put into it, it is still Stan Ockers' program. He should be congratulated on doing such a fine job and making my task such a pleasure.

Program Flow

Doggies begins by jumping to a massive block of initialization code which is placed at the end of the program so as not to slow things down when the main program is executing later on.

I believe the first thing you should do in **any** program is to have something happen immediately after typing RUN, even if you only clear the screen and print the title. A pause of 2 or 3 or even 20 seconds before anything happens is totally unacceptable, as the user may think the program has crashed. You should also make no assumptions as to screen conditions before the program was run. Therefore, as a rule of thumb, always start the initialization with a GRAPHICS statement, followed by the necessary POKES and SETCOLOR statements to set margins, colors and so on, even if you are using default values. Doggies also clears out and reserves an area of memory at the top of RAM (see Memory Allocation), then prints the instructions and the word "INITIALIZING" so you know what's going on. You can now read the instructions oblivious of the initialization which continues as you read. This takes about 11 seconds. When finished, "INITIALIZING" is overprinted by another message to "Press START to begin". When you've done so, the screen will be cleared, player missile graphics are enabled and the vertical blank interrupt routine is set up. The BREAK key is also disabled to force you to use SYSTEM RESET to abort the program. This is the only way to ensure all system parameters are back to normal when you return to BASIC. Whew!

We now set up the screen for a new game and cycle through the main program loop from lines 50 to 80. Yes, that's right! The main program loop is a mere 4 lines long! The remainder of the program is subroutines for specific actions. Each subroutine commences on a line number of a multiple of 100 and is preceded by a REM to indicate its function. I will delve into some of these later. The remainder are fairly straightforward.

Doggies uses a number of the ATARI's special features such as interrupt processing, player-missile graphics and a redefined character set. We'll now take a look at some of these aspects, but be warned: A good working understanding of BASIC is assumed.

Memory Allocation

As a lot of our members are beginners with minimum systems, I intended right from the start for Doggies to run on a cassette based system with only 16k RAM. It does this quite admirably, but if you've got a disc based system, you'll need to delete all the REMarks if you expect to run it in 16k. (You shouldn't really be using a disc drive with only 16k anyhow, as there is just too little room left for programs or high resolution graphics modes.)

Figure 1 shows the maximum memory requirements for Doggies when typed in exactly as shown in the listing. RAMTOP is an Operating System pointer at memory location 106 (\$6A) which tells us the page number of the first non-RAM byte in memory. (A page is a block of 256 bytes). By POKing a lower value into RAMTOP, we can fool the system into thinking it has less RAM than it actually has. When we carry out a GRAPHICS command, the display memory and display list will be written below the new value of RAMTOP, thereby reserving an area for our own use. We use this technique in Line 2000 to set aside an area for player-missile graphics and the redefined character set.

The player-missile graphics require 4 pages (or 1024 bytes) of memory for double line resolution and must start on a 1k boundary. We define the starting location with the variable START. The first 384 bytes of this area (i.e. START to START + 383) are not used. The area from START + 384 to START + 511 is reserved for missiles, but we will not be using them. It seems a shame to waste 512 bytes, so seeing the redefined character set requires 512 bytes, we can store it in this area. We also need a 1 page buffer between the display memory and START due to the RAMTOP dragon who gobbles up the first 64 bytes above RAMTOP whenever a GRAPHICS command or clear screen command are executed.

The GRAPHICS 0 display used for the instructions is shown in figure 1 as it is more memory intensive than the GRAPHICS 18 display used for the game itself. Just out of interest, GRAPHICS 18 only requires 20 bytes for its display list and 240 bytes for its display memory.

You can see from Figure 1 the BASIC program itself requires 7245 bytes just to reside in memory. (The structure of the tokenized BASIC program is not important at this point.) When run, it will then take up an extra 61 bytes for string storage plus a variable amount for the run-time stack (this keeps track of return addresses for GOSUBs and FOR...NEXT loops.) The area labelled "DOS" is only applicable if you've got a disc drive booted. The figures are for DOS 2. The area labelled "Operating System RAM" is reserved by the Operating System for a variety of functions. (Maybe we could delve into all this stuff in future issue.) By adding up the number of bytes used by your system, you can see a cassette based system requires well under 16k (note 16k is 16384 bytes), whereas a disc based system requires just over 16k, hence the need to delete the REMarks for a disc based system with only 16k.

Player-Missile Graphics

Whenever using player-missile graphics, you must first reserve an area (as discussed above) to store the images and then clear this area of any extraneous garbage. I used a little trick to accomplish the latter by issuing a GRAPHICS 21 command prior to reserving the P-M graphics area (see Line 2000). This has the effect of clearing out 960 bytes at the top of memory for the GRAPHICS 21 display. When RAMTOP is moved down, the 640 bytes used by P-M graphics remains undisturbed and set to all zeros. GRAPHICS 0 or 5 could also be used to clear out the same amount of memory, but these may have caused a disturbing flash of blue from the background color of the text portion of their displays. Care to minor details like this makes a really professional program.

Only Player 0 is used (for the bone) and this is double width. Its shape is read directly into the Player 0 area (START + 512 to START + 639) at Line 2140.

Vertical Blank Interrupt Processing

The vector to the Vertical Blank Interrupt (VBI) service routine is set by the call X = USR(1536) as the very last function of the initialization code. See the assembled source code for more details. For further information on how a VBI works, see "Flashing Cursor" in the December 1982 issue of INSIDE INFO. x Once set up, the VBI reads the joystick to determine whether it has been pushed left or right and moves the bone accordingly. As this is done 50 times per second regardless of what else is happening in the program, it results in a beautiful smooth motion. Again refer to the assembled source code for more details.

Redefined Character Set

Two pages of the character set in ROM (i.e. 64 characters or 512 bytes) are copied to the new locations in our reserved area. This process is speeded up by utilizing a machine language routine stored in ML\$, (Refer to the assembled source code to see how it works.) It was stored in a string so you can easily incorporate it into any of your own programs. The general form of the call is X = USR(ADR(ML\$), 57344, CH). 57344 is the address of the start of the character set in ROM. Change this to 57856 if you wish to copy the second half of the character set (i.e. lower case letters and control graphics characters). CH is the address of the start of your new character set. You may also use this routine to copy the whole of the character set (i.e. 128 characters or 1020 bytes) by changing the last number of the DATA statement in Line 2170 from 2 to 4. This indicates how many pages are to be copied.

The Operating System pointer CHBAS at location 756 (\$2F4) tells ANTIC where the character set starts. Note we do not change this until after the instructions are cleared from the screen, otherwise some of the letters change into parts of little doggies while we try to read them. Very annoying!

When the character set has been copied, we redefine 34 of the 64 characters. This is the most time consuming part of the initialization. Allocating which characters are to be changed is quite a challenge, as the doggies require 34 characters, the score requires 10, and the titles and various messages I want to print require about 24 characters — a total of 68 characters, but only 64 characters are in a GRAPHICS 2 character set. I couldn't delete any parts of the doggies or the digits for the score, but by carefully rewording the messages and re-allocating some of the characters for the doggies, I was able to reach an acceptable compromise. Hence messages like "GREAT STUFF" instead of "EXCELLENT", as the "X" became part of a doggie. Every character except the comma is used at some time somewhere in the screen display.

When the new characters are put together in the correct pattern, they will form the shapes for the various doggies. The shape of each doggie is made up of 6 characters in a 2 by 3 grid. As an example, the stationary doggie is shown in Figure 2. The characters for each shape are stored in DATA statements. Lines 1000-1013 are for the white doggies and Lines 1100-1113 are for the brown doggies. There must be at least 6 characters in each DATA statement, so the trailing blank spaces in Lines 1010-1013 are replaced by inverse blanks. This is important. The program will crash and you'll be given an error message otherwise. The DATA statements in Lines 1100-1113 are exactly the same as Lines 1000-1013, but in inverse video. This is how we get the 2 different colored doggies. A major subroutine of the program occurs at Lines 200-220. It prints the string DOG\$ at POSITION X,Y. DOG\$ is determined by the variable LINE. By changing LINE, we can change the shape of the printed doggie.

The current position of the doggies is stored in P\$ (1 for a white doggie, 2 for a brown doggie, and 0 for the blank space). By comparing P\$ with the final position represented in F\$ (Line 440), we can determine if he end has been reached.

Attract Mode

I could say Doggies is constantly in attract mode, as something is always happening irrespective of what the user is doing. Every time through the main loop, the program checks to see if the fire button has been pressed. If it hasn't, then it randomly selects a doggie and moves him in accordance with one of 3 randomly selected subroutines. These have the effect of making him bark, wag his tail and stomp his feet. Even when the game is over, the attract mode continues while waiting for you to press START for a new game.

The ability to move a doggie when the game is over might cause havoc. This is averted by setting a flag called ATTRACT. When ATTRACT is 0, the game is in progress and the GOTO in Line 80 makes sure the fire button is checked. When the game is over, ATTRACT is set to 10 so Line 80 will jump past the fire button routine.

Colors

The display uses a total of 6 colors in a very pleasing combination. The black background is colored by the background color register and the scungy yellow bone is colored by the Player 0 color register. Playfields 0 and 2 are used for the white doggies and brown doggies respectively. Playfields 1 and 3 are used for the green writing and blue writing respectively (you won't see the blue until the game is over). Green writing can therefore be achieved by printing a message in lower case and blue writing can be achieved by printing in inverse lower case. A problem arises here when we wish to print the score, as the digits 0 to 9 are in the range of characters colored by Playfield 0, but we want them to be colored by Playfield 3. We therefore need to convert the score to a string and manipulate the individual digits to change their color from white to green. This is done in Line 430.

The colors show up fairly well on a black and white TV and the only evidence of color "bleeding" is from the brown doggies. This is an unavoidable problem common with dark luminances. It can only be overcome by careful selection of color combinations.

Sounds

BARX\$ contains the range of tones for the doggies' cute barking sound used in Line 500. This was one point with which Stan Ockers was not pleased, but I could not come up with anything better. Apart from which, I quite like it.

The subroutine for the footsteps sound in Lines 900-910 is unusual. The odd numbered distortion value causes the speaker to click, then silence. Turning the voice off then causes another click. When executed together, the 2 clicks are indistinguishable and sound as though they are combined to form one single louder click. Prior to entering this subroutine, the variables V and INC are specified. V is the volume of the footsteps and INC is the increment by which the volume is increased or decreased. In this way the footsteps may stay at a constant volume while the doggie is stationary (INC = 0), decrease as the doggie walks away (INC = -1) or increase as the doggie approaches (INC = 1). Using a common subroutine ensures the timing of the footsteps is always the same. This subroutine is also the reason for the apparent duplication of DATA statements at Lines 1010-1013 and 1110-1113.

Scoring

Your score is incremented every time a doggie is moved. The object is to move all the doggies in as few moves as possible. 15 moves is the best you can do, but scores of around 21 are more common.

There is a sort of bug in the scoring routine. Even though it will never be encountered under normal use, I mention it here as a good example of deciding where to "draw the line" for certain error conditions. The score's color conversion mentioned above will crash if there are more than 3 digits in the score. I decided on a maximum of 3 digits for 2 reasons. Firstly, more than 3 digits will wraparound onto the next line and mess up the display. Secondly, you will have to be a complete moron to need 1000 moves. Even the smart alec who intentionally tries to crash the program will have to move one doggie every 5 seconds for an hour and a half before the program will complain. If the idea of this bug still worries you, just increase the size of MOVES in Line 2070 to 4. Our hypothetical smart alec will then be moving doggies for 14 hours!

Human Engineering

One of the most important aspects of any program is its human engineering. Here is where Doggies excels. It couldn't possibly be any simpler to use! After initialization, you need only press START for the game to begin. In fact, you can then press START again at any time and the game will restart. This comes in handy if you can see you've made a mistake and don't want to carry the game through to completion knowing you'll get a bad score.

The bone is controlled by pushing left or right on a joystick plugged into Port 1. If it goes off the screen, it will wrap around to the other side. A doggie is selected by placing the bone under the doggie you wish to move and pressing the fire button. The bone is not very fussy when it comes to selecting a doggie. It doesn't have to be directly under a doggie, just make sure at least half of it is under the doggie you wish to move. If it is exactly in the middle of 2 doggies, a choice still has to be made, so it opts for the doggie on the right.

The program won't let you make an illegal move. I won't spoil the surprise by telling you what happens, so try it and see for yourself.

As there is no keyboard input, the well known random color switching normally comes into effect after just under 11 minutes. As Doggies is such a compulsive game, you will quite likely be playing for well over 11 minutes, so the random color switching must be catered for. Any legal input (i.e. pressing START, pushing the joystick left or right or pressing the fire button) will reset the ATTRACT flag at location 77 (\$4D), thereby avoiding the random color switching. If however you have to answer the 'phone or you go on holidays and forget to turn the computer off, the random color switching will still be enabled as usual. This will protect the phosphors in the TV set. When you come back, just move the joystick, press the fire button or press START and everything will return to normal.

All in all, the human engineering is so well done even the youngest child (or the oldest computer critic) can also enjoy playing Doggies.

Conclusion Well, that about wraps it up for Doggies. If you've read this far and haven't keyed it in yet, then you're missing out on quite a treat. Doggies is undoubtedly the most professional game we've printed so far, so give it a go. We'll publish the solution next issue.



HIDDEN ATARI DOS FUNCTIONS

by SHANE ROLIN OF C SOFTWARE

1ST IN A SERIES

PART #1

What happens when you can't find your Basic cartridge, and really you want to type in that new program listing you found? Well, don't worry because there is a way to type in Basic and maybe even another language through Atari Dos.

In order to do this, first, go to Dos and when the menu appears, type a "C" and RETURN. You will then be prompted with "COPY-FROM,TO?". Enter the following "E:D:\filename" and RETURN. You will hear the disk drive go on and open a file to the filename entered. The drive light will go out soon after it goes on leaving the cursor one line below the last line typed.

You can now type in a Basic program just as though you were in Basic. At the end of each line hit RETURN just as in Basic. The only limit to the length of the program you type in is the memory of your computer. Be careful about mistakes because the only way you can correct them is by typing in the line again. So proofread carefully before you hit the RETURN key.

I think you will find this an interesting function of Atari Dos. You are not limited to Basic programs. You may also type in data for programs or a message for a friend. To stop this function hit the CTRL key and 3 at the same time and your program will be "LISTED" out to disk. Do not refrain from using inverse characters, lower case or control characters; the Editor will accept anything you type into it. To read back messages, simply hit "C" and "filename,E:" and your message will be loaded into memory and be displayed. To stop the scrolling hit CTRL 1; hit it again to continue. To load a program typed in through Dos, simply go to Basic and "ENTER "D:\filename".

PART #2

You're in Dos and you want to reboot the system or just return to Basic. One way is to hit the "B" function of Dos, which in "RUN CARTRIDGE", will exit to the cartridge you have in the left slot. Another way is through the "M" function of Dos "RUN AT ADDRESS". Select this function, enter F125 or E477 when the computer asks "RUN FROM WHAT ADDRESS" and hit RETURN. The computer will do a "COLDSTART", the same way as if you shut the machine off and turn it back on again. When you do this, all memory locations will be cleared and the computer will execute the cartridge if you have one in. If you want to do a "WARMSTART", instead of entering "F125", enter F128 or E474 and the computer will perform a "WARMSTART", the same as hitting the "SYSTEM RESET" key, and execute the left cartridge.

MESSAGE READER IS A UTILITY TO READ ALL LISTED FILES OFF OF THE DISK DRIVE AND DISPLAY THEM TO THE SCREEN OR PRINTER.

—SHANE ROLIN OF P.A.C.E.

DOGGIES Revisited

```

1 REM #####
2 REM # DOGGIES #
3 REM # by Stan Ockers #
4 REM # Eugene A.C.E. Newsletter #
5 REM # January 1982 #
6 REM # Modified by Garry Francis #
7 REM # Reprinted by A.C.E.(N.S.W.) #
8 REM # February 1983 #
9 REM #####
10 GOSUB 2000
19 REM ### Main loop ###
20 POKE 77,0: ? &6;CHR$(125);" dog
gies"? &6;? &6;" number of moves "&CH
R$(16)
30 P$="1110222":FOR C=1 TO 7:GOSUB 100
:IF A THEN GOSUB 210
40 NEXT C:POKE 209,120:POKE 53248,120:
MOVE=0:ATTRACT=0
50 IF STRIG(0)=0 THEN GOSUB 300:GOTO 5
0
60 IF PEEK(53279)=6 THEN POKE 209,0:PO
KE 53248,0:GOTO 20
70 C=INT(7*AND(0))+1:GOSUB 100:IF A TH
EN GOSUB 500+100*INT(3*AND(0))
80 GOTO 50+ATTRACT
99 REM ### Which doggie? ###
100 A=VAL(P$(C,C)):IF A=1 THEN LINE=10
00
110 IF A=2 THEN LINE=1100
120 X=30C-3:RETURN
199 REM ### Draw doggie ###
200 FOR J=1 TO 14:NEXT J
210 RESTORE LINE:READ DOG$:POSITION X,
6: ? &6;DOG$(1,2)
220 POSITION X,7: ? &6;DOG$(3,4):POSITI
ON X,8: ? &6;DOG$(5,6):RETURN
299 REM ### Process player's move ###
300 POKE 77,0:B=0:C=INT(POKE(209)/24-
.5):IF C<1 THEN C=1
310 IF C>7 THEN C=7
320 B=B+1:IF VAL(P$(B,B)) THEN 320
330 IF B=C THEN RETURN
340 IF C*B-2 OR C*B+2 THEN GOSUB 100:G
OSUB 800:RETURN
350 TEMP=C:FOR C=1 TO 7:IF C=TEMP THEN
400
360 GOSUB 100:IF A=0 THEN 400
370 IF C<TEMP THEN LINE=LINE+6
380 IF C>TEMP THEN LINE=LINE+7
390 GOSUB 210
400 NEXT C:C=TEMP:GOSUB 100:GOSUB 500:
GOSUB 600:LINE=LINE+8:V=8:INC=0:GOSUB
900:LINE=LINE+2:INC=-1:GOSUB 900
410 LINE=LINE+2:GOSUB 900:POSITION X,6
: ? &6;" "IP$(B,B)=P$(C,C):IP$(C,C)=0"
: C=B
420 GOSUB 100:LINE=LINE+12:V=1:INC=1:G
OSUB 900:LINE=LINE-2:GOSUB 900:LINE=LI
NE-10:GOSUB 700:GOSUB 600
430 MOVE=MOVE+1:MOVE$=STR$(MOVE):POSITI
ON 17,2:FOR I=1 TO LEN(MOVE$): ? &6;CH
R$(ASC(MOVE$(I,I))-32):NEXT I
440 IF P$(OF$ THEN RETURN
450 POSITION 2,4:IF MOVE=15 THEN ? * &6;
" great stuff":GOTO 480
460 IF MOVE<20 THEN ? &6;" good goin
g":GOTO 480
470 ? &6;"could be better"
480 POSITION 0,11: ? &6;"press start to
begin":ATTRACT=10:RETURN
499 REM ### Bark ###
500 LINE=LINE+5:GOSUB 210:FOR I=1 TO 6
: SOUND 0,ASC(BARK$(I)),12,14-I*2:SOUND
1,ASC(BARK$(I)),14,I*2:NEXT I
510 LINE=LINE+5:GOSUB 210:SOUND 0,0,0,
0:SOUND 1,0,0,0:RETURN

```

```

599 REM *** Wag tail ***
600 LINE=LINE+1:FOR I=1 TO 3:LINE=LINE
+1:GOSUB 200:LINE=LINE-1:GOSUB 200:NEXT
I:LINE=LINE-1:GOSUB 200:RETURN
699 REM *** Hop ***
700 LINE=LINE+3:V=8:INC=0:GOSUB 900:LI
NE=LINE-3:GOSUB 200:RETURN
799 REM *** Shake head ***
800 FOR I=1 TO 3:LINE=LINE+6:GOSUB 200
:LINE=LINE-6:GOSUB 200
810 LINE=LINE+7:GOSUB 200:LINE=LINE-7:
GOSUB 200:NEXT I:RETURN
899 REM *** Footsteps sound ***
900 FOR I=1 TO 3:V=V+INC:LINE=LINE+1:G
OSUB 200:SOUND 0,6,13,V:SOUND 0,0,0
910 LINE=LINE-1:GOSUB 200:SOUND 0,11,1
3,V:SOUND 0,0,0,0:NEXT I:RETURN
999 REM *** Shapes of doggies ***
1000 DATA '(')*XZ8
1001 DATA '(')\XZ8
1002 DATA '('[XZ8
1003 DATA '('[XZ-
1004 DATA '(')\+8
1005 DATA '!'*XZ8
1006 DATA '!\XZ8
1007 DATA MY<=XZ8
1008 DATA >?Z+8
1009 DATA >?]\Z-
1010 DATA @HJK
1011 DATA @HJK
1012 DATA XQ
1013 DATA XQ
1100 DATA '(')*XZ8
1101 DATA '(')\XZ8
1102 DATA '('[XZ8
1103 DATA '('[XZ-
1104 DATA '(')\+8
1105 DATA '!'*XZ8
1106 DATA '!\XZ8
1107 DATA MY<=XZ8
1108 DATA >?Z+8
1109 DATA >?]\Z-
1110 DATA @HJK
1111 DATA @HJK
1112 DATA XQ
1113 DATA XQ
1999 REM *** Initialisation ***
2000 GRAPHICS 21:START=PEEK(106)-4:POKE
E 106,START-1:GRAPHICS 0:POKE 710,0:PO
KE 709,12:POKE 752,1:POKE 82,1
2009 REM *** Instructions ***
2010 POSITION 16,1:"DOGGIES"? :? "Y
here are 3 white doggies on the left"
2020 ? "side of the screen and 3 brown
doggies"? : "on the right side. You m
ust reverse"
2030 ? "their position by moving one d
oggie at"? : "a time."? :? "Use the jo
ystick in Port 1 to put the"
2040 ? "bone under the doggie you wish
to"? : "move, then press the fire butt
on. The"
2050 ? "doggie will move into the empt
y space."? : "but only if he is next to
it or no"
2060 ? "more than one doggie away."? :
? "It can be done in 15 moves! Can y
ou"? : "do it?"
2070 POSITION 13,22: ? " INITIALISING "
:DI1 BARK$(6),ML$(32),DOGS$(6),MOVES$(3)
,F$(7),P$(7):F$="2220111"
2079 REM *** Data for barking ***
2080 RESTORE 2090:FOR I=1 TO 6:READ A:
BARK$(I)=CHR$(A):NEXT I
2090 DATA 97,109,97,9,5,5

```

```

2099 REM *** UBI service routine ***
2100 FOR I=1536 TO 1578:READ A:POKE I,
A:NEXT I
2110 DATA 104,160,10,162,6,169,7,76,92
,228,173,0,211,72,41,8,208,6,230,209,1
04,24,144,7,104,41,4,208,11,198,209
2120 DATA 169,0,133,77,165,209,141,0,2
08,76,98,228
2129 REM *** P-M Graphics ***
2130 POKE 54279,START:POKE 53256,1:PM=
256*START:POKE 53248,0:POKE 209,0:POKE
704,30
2140 FOR I=PM+602 TO PM+604:READ A:POK
E I,A:NEXT I
2150 DATA 195,255,195
2159 REM *** Move character set ***
2160 FOR I=1 TO 32:READ A:ML$(I)=CHR$(
A):INEXT I:ICH=256*START:I=XUSR(ADR(ML$),
57344,CH)
2170 DATA 104,104,133,204,104,133,203,
104,133,206,184,133,205,162,2
2180 DATA 160,0,177,203,145,205,136,20
8,249,230,204,230,206,202,208,240,96
2189 REM *** Redefine characters ***
2190 READ X:IF X=-1 THEN 2560
2200 FOR I=CHX TO CH+X-7:READ A:POKE
I,A:NEXT I:GOTO 2190
2210 DATA 8,7,15,31,61,109,111,110,111
2220 DATA 16,224,240,248,188,182,246,1
18,246
2230 DATA 24,12,12,7,3,3,7,15,31
2240 DATA 32,48,224,192,192,224,240
,248
2250 DATA 40,31,31,31,31,31,24,248,248
2260 DATA 48,248,248,248,248,248,24,31
,31
2270 DATA 56,0,7,31,63,125,237,239,206
2280 DATA 64,0,224,248,252,198,183,247
,115
2290 DATA 72,15,12,15,7,3,7,15,31
2300 DATA 80,240,48,240,224,192,224,24
0,248
2310 DATA 88,31,31,31,31,27,248,248,0
2320 DATA 104,248,248,248,248,216,31,3
1,0
2330 DATA 112,0,7,15,31,63,63,55,55
2340 DATA 120,0,128,192,192,64,127,255
,255
2350 DATA 208,55,7,7,3,3,7,15,31
2360 DATA 216,254,224,254,252,192,224
,240,248
2370 DATA 224,127,7,127,63,3,7,15,31
2380 DATA 232,236,224,224,192,192,224
,240,248
2390 DATA 240,0,7,31,63,127,239,239,20
7
2400 DATA 248,0,224,248,252,254,247,24
7,243
2410 DATA 256,1,7,15,11,3,1,3,7
2420 DATA 320,128,224,240,208,192,128,
192,224
2430 DATA 336,7,7,7,7,7,4,28,0
2440 DATA 344,224,224,224,224,224,32,5
6,0
2450 DATA 392,192,192,128,192,192,192,
192,96
2460 DATA 440,0,1,3,3,2,254,255,255
2470 DATA 448,3,3,1,3,3,3,6
2480 DATA 456,0,224,240,248,252,252,23
6,236
2490 DATA 464,15,15,15,7,7,15,31
2500 DATA 472,15,12,15,71,67,71,47,31
2510 DATA 480,240,48,240,226,194,226,2
44,248
2520 DATA 488,15,15,15,71,67,71,47,31

```



```

00010 .LI OFF
00020 OR #5E00
00030 *****
00040 * UN-COMPACTOR *
00050 * NEWS LETTER ARTICLE (PART3) *
00060 * BY TOM NEWMAN *
00070 * DECEMBER 1982 *
00080 *****
00090 *
00100 *****
00110 * EQUATE TABLE *
00120 *****
00130 BUFFER .EQ #F0
00140 BUFFLO .EQ #F0
00150 BUFFHI .EQ #F1
00160 TABLE .EQ #F2
00170 TBLLO .EQ #F2
00180 TBLHI .EQ #F3
00190 SAVEY .EQ #F4
00200 REPEATS .EQ #F5
00210 HOMELO .EQ #F6
00220 HOMEHI .EQ #F7
00230 DATA .EQ #F8
00240 *****
00250 * INITIALIZATION *
00260 *****
00270 START LDA #50
00280 STA BUFFLO
00290 STA HOMELO
00300 LDA #61
00310 STA BUFFHI
00320 STA HOMEHI
00330 LDA #00
00340 STA TBLLO
00350 STA SAVEY
00360 LDA #40
00370 STA TBLHI
00380 LDX #C0
00390 *****
00400 * PROGRAM BODY *
00410 *****
00420 CHANGE LDY #00
00430 LDA (TABLE),Y
00440 STA DATA
00450 INY
00460 LDA (TABLE),Y
00470 STA REPEATS
00480 CLC
00490 LDA TBLLO
00500 ADC #02
00510 STA TBLLO
00520 LDA TBLHI
00530 ADC #00
00540 STA TBLHI
00550 BLOCK LDY SAVEY
00560 LDA DATA
00570 STA (BUFFER),Y
00580 CLC
00590 LDA BUFFLO
00600 ADC #28
00610 STA BUFFLO
00620 LDA BUFFHI
00630 ADC #00
00640 STA BUFFHI
00650 DEX
00660 BEQ NEXTCOLM
00670 REENTER INC REPEATS
00680 BNE BLOCK
00690 JMP CHANGE
00700 NEXTCOLM INC SAVEY
00710 CPY #27
00720 BEQ OUT
00730 LDA HOMELO
00740 STA BUFFLO
00750 LDA HOMEHI
00760 STA BUFFHI
00770 LDX #C0
00780 JMP REENTER
00790 OUT BRK
00800 .END

```

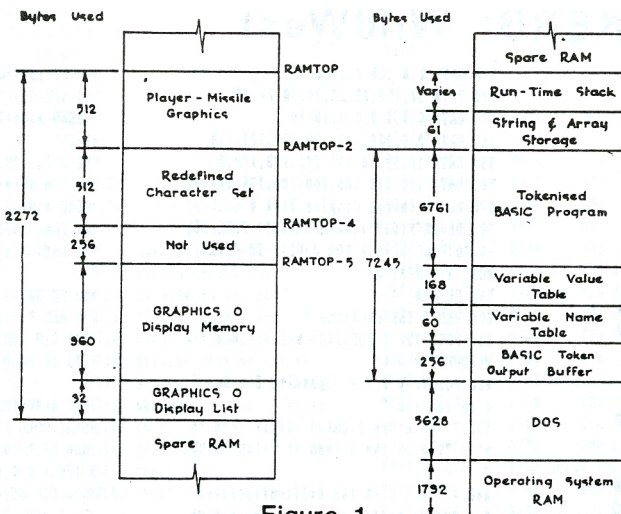


Figure 1

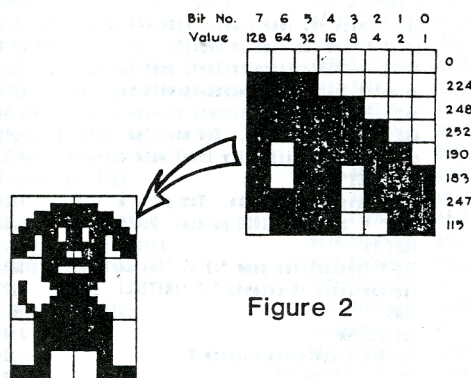


Figure 2

```

*****
10 REM * A FREDERICK ATARI COMPUTER *
   * TECHNICAL SOCIETY ORIGINAL *
   * PROGRAM *
20 REM REMOVER by Robert Schniebolk
   REMOVER processes files which
   have been created by REMAKER to
30 REM remove the line numbers and REM
   statements produced by REMAKER.
   It's output returns the file to
40 REM its original form after editing
   is completed. REMOVER's output
   file name is the same as the
50 REM source file, but does not have
   an extension. .... RJS 01/09/83
60 POKE 82,0:GRAPHICS 0:?"REMOVER-by
   Bob Schniebolk":? :? :? :?
70 DIM A$(120),I$(15),O$(11)
80 CLOSE #1:CLOSE #2:?"ENTER INPUT FI
   LESPEC ":INPUT I$:?
90 IF I$(LEN(I$)-2)()="REM" THEN ? CHR$
   (253):"FILE MUST HAVE 'REM' EXTENTION
   PRODUCED BY REMAKER!":? :GOTO 80
100 D$=I$(1,LEN(I$)-4)
110 TRAP 80:OPEN #1,4,0,I$:OPEN #2,8,0
   ,0$
120 ? :? "PROCESSING FILE":? :TRAP 150
130 INPUT #1,A$:IF LEN(A$)>5 THEN ? #2
   ,A$(6):? A$(6)
140 GOTO 130
150 CLOSE #1:CLOSE #2:?"CHR$(253)":?"T
   ASK COMPLETED":TRAP 200

```


OCKERS: WildWest

```

1 REM *****
2 REM **ATARI COMPUTER ENTHUSIASTS**
3 REM *****NEWSLETTER*****
4 REM **      3662 Vine Maple DR      **
5 REM **      EUGENE OR 97405      **
6 REM **      April 83      **
7 REM **      10 yr      **
8 REM *****
9 REM **WildWest
  **
10 REM *by Stan Ockers*
140 GOSUB 450:GRAPHICS 0:POKE 756,CSPA
GE:GOSUB 610:GOSUB 860:GOSUB 1000:GOSUB
B 1110:GOSUB 1330
150 ? "Press START to begin"
160 IF PEEK(53279)<6 THEN 160
170 GOSUB 580:RESTORE 180:FOR J=704 TO
712:READ A:POKE J,A:NEXT J:BKG=56
180 DATA 0,44,92,34,66,14,50,0,56
190 DIF=1:SCORE=0:HATS=4:CHR(125):P
OSITION 21,0:?"dif score high":PO
KE 1761,100:POKE 1762,100
200 FOR J=53248 TO 53251:POKE J,100:NE
XT J:POKE 1763,2:POKE 1766,200:POKE 17
67,40:GOSUB 1370:BONUS=1000
210 Y=20:FOR X=3 TO HATS*3 STEP 3:GOSU
B 840:NEXT X:POSITION 14,0:?"HIGH-A:US
R(1536):POKE 559,46:POKE 53277,3
220 IF PEEK(53279)=5 THEN DIF=DIF+1:IF
DIF=10 THEN DIF=1
230 POSITION 2,0:?"DIF:FOR J=1 TO 100:
NEXT J:IF PTRIG(0)=1 THEN 220
240 GOSUB 1370:POKE 1760,0:POKE 1781,0
:POKE 1788,0:POKE 77,0
250 IF RND(0)<0.01*?DIF THEN POKE 1780,
1
260 INCR=SCORE+PEEK(1768)*5:POSITION 6
,0:?"INCR
270 IF PEEK(1760)=0 THEN 250
280 SCORE=INCR:SOUND 1,0,0,0
290 IF SCORE>BONUS THEN BONUS=BONUS+10
00:IF HATS<9 THEN HATS=HATS+1:Y=20:X=3
:HATS:GOSUB 840
300 IF PEEK(1768)<(PEEK(1769)) THEN GOSU
B 730:GOTO 320
310 DIF=DIF+1:IF DIF=9 THEN DIF=9
320 IF HATS=0 THEN 350
330 GOTO 220
340 REM * game over routine *
350 POSITION 1,7:?" 00 0 0 00 00
00 0 00 00"
360 POSITION 1,8:?" 0 0 00 00 00 0
0 0 0 0 0 0"
370 POSITION 1,9:?" 0 0 00 0 00 0
0 0 00 00 0 0"
380 POSITION 1,10:?" 0 00 000 0 0
0 0 00 0 000"
390 POSITION 1,11:?" 0 0 00 0 0 0
0 0 000 0 0 0"
400 POSITION 1,12:?" 00 0 00 0 00 0
00 0 00 0 0"
410 IF SCORE>HIGH THEN HIGH=SCORE
420 IF PEEK(53279)<6 THEN 420
430 GOTO 190
440 REM * change character set *
450 DIM MCS$(42):RESTORE 460:FOR J=1 T
O 42:READ A:MCS$(J,J)=CHR$(A):NEXT J
460 DATA 104,169,0,133,203,133,205,169
,224,133,204,165,106,56,233,5,133,106,
24
470 DATA 105,1,133,206,162,4,160,0,177
,203,145,205,200,208,249,230,204,230,2
06,202,208,242,96
480 A=USR(ADR(MCS$)):CSPAGE=PEEK(106)+
1:CS=256+CSPAGE
490 RESTORE 500:FOR J=CS+8 TO CS+63:RE
AD A:POKE J,A:NEXT J:RETURN
500 DATA 128,2,32,1,134,1,32,8

```

```

510 DATA 2,8,128,2,64,8,32,2
520 DATA 32,130,12,28,20,20,20,20
530 DATA 0,0,8,9,9,9,10,10
540 DATA 0,0,128,128,128,128,128,128
550 DATA 10,10,10,143,143,138,170,0
560 DATA 128,128,128,200,200,136,168,0
570 REM * change display list *
580 DL=PEEK(560)+256*PEEK(561):POKE DL
+3,70:POKE DL+6,6:FOR J=DL+7 TO DL+28:
POKE J,4:NEXT J
590 RETURN
600 REM * instructions *
610 POKE 752,1:POSITION 5,1:?" * * *
WILDWEST * * * "
620 POSITION 2,3:?"Dynamite Dan has i
t in for you."
630 ? "He drops lighted sticks from th
e":?"top of the screen at rates which
"
640 ? "vary with the difficulty level.
":?"Using paddle zero you move a somb
ero"
650 ? "to catch them before they reach
the":?"bottom and explode. Each tim
e you"
660 ? "miss you lose a hat. Lose all
hats":?"and the game is over."
670 ? :?"The difficulty level goes do
wn on":?"each miss, increases with ea
ch"
680 ? "successful group. You may also
change":?"the difficulty level with t
he SELECT"
690 ? "key during breaks. You get a b
onus":?"hat every 1000 points. Use S
TART to"
700 ? "restart the game.":?"Initial
ization takes 18 seconds.":?"INITIALI
ZING"
710 RETURN
720 REM * explosion routine *
730 X=0:Y=0:COL=20
740 IF PEEK(1664+X)=0 THEN 780
750 P=PEEK(1724+X)+PEEK(1736+X)*256:PO
KE P,1:POKE P+1,2:J=J+1:IF J=4 THEN J=
0
760 FOR K=0 TO 2:SOUND J,50+RND(0)*50,
8,13+K:NEXT K:POKE 712,COL+8*X
770 POKE P,PEEK(1712+X):POKE P+1,0:FOR
L=1 TO 30:RND(0):NEXT L
780 X=X+1:IF X<12 THEN 740
790 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUND
2,0,0,0:SOUND 3,0,0,0:POKE 712,BKG
800 X=HATS*3:Y=20:POSITION X,Y:?" "
POSITION X,Y+1:?" "HATS=HATS-1
810 DIF=DIF+1:IF DIF=0 THEN DIF=1
820 RETURN
830 REM * print hat *
840 POSITION X,Y:?"%":POSITION X,Y+1
:?"&":RETURN
850 REM * PM graphics *
860 DIM X$(1):A=ADR(X$):B=INT((A-512)/
1024+1)*1024:DIM F$(B-A+511):DIM P0$(1
28)
870 DIM P1$(128),P2$(128),P3$(128):POK
E 54279,B/256
880 DIM C1$(15):RESTORE 890:FOR J=1 TO
15:READ A:C1$(J,J)=CHR$(A):NEXT J
890 DATA 16,56,186,124,0,40,0,40,16,19
8,170,146,130,68,68
900 P0$(1)=CHR$(0):P0$(128)=CHR$(0):P0
$(2)=P0$:P1$=P0$:P2$=P0$:P3$=P0$:P0$(2
6)=C1$
910 DIM C2$(11):RESTORE 920:FOR J=1 TO
11:READ A:C2$(J,J)=CHR$(A):NEXT J
920 DATA 124,254,254,124,56,16,0,0,0,1
98,130

```

```

930 P2$(30)=C2$
940 DIM C3$(6):RESTORE 950:FOR J=1 TO
6:READ A:C3$(J,J)=CHR$(A):NEXT J:P3$(3
4)=C3$
950 DATA 170,184,170,184,170,184
960 DIM H$(9):RESTORE 970:FOR J=1 TO 9
:READ A:H$(J,J)=CHR$(A):NEXT J:P1$(80)
=H$:POKE 53257,1
970 DATA 124,198,124,124,56,56,56,56,4
0
980 RETURN
990 REM * various strings *
1000 DIM CSWD$(15):RESTORE 1020:FOR J=
1 TO 15:READ A:CSWD$(J,J)=CHR$(A):NEXT
J
1010 A=ADR(CSWD$):HI=INT(A/256):LO=A-2
56*HI:POKE 1774,LO:POKE 1776,HI
1020 DATA 30,142,1,25,140,1,20,138,1,1
5,138,1,0,0,0
1030 DIM DSPD$(9):RESTORE 1040:FOR J=1
TO 9:READ A:DSPD$(J,J)=CHR$(A):NEXT J
1040 DATA 5,5,4,4,4,3,3,2,2,1,1
1050 DIM DDLY$(9):RESTORE 1060:FOR J=1
TO 9:READ A:DDLY$(J,J)=CHR$(A):NEXT J
1060 DATA 30,25,20,15,10,5,5,5,5
1070 DIM CNT$(9):RESTORE 1080:FOR J=1
TO 9:READ A:CNT$(J,J)=CHR$(A):NEXT J
1080 DATA 15,20,25,30,35,40,45,50,55
1090 RETURN
1100 REM * create VBI string *
1110 DIM VBI$(398):RESTORE 1120:FOR J=
1 TO 398:READ A:VBI$(J,J)=CHR$(A):NEXT
J:RETURN
1120 DATA 173,234,6,240,57,206,236,6,1
6,52,173,238,6,133,208,173,240,6,133,2
09,172,242,6
1130 DATA 177,208,240,21,141,0,210,200
,177,208,141,1,210,200,177,208,141,236
,6,200,140,242,6
1140 DATA 208,14,169,0,141,0,210,141,1
,210,141,234,6,141,242,6,216,173,224,6
,240,3,76,98,228
1150 DATA 173,112,2,73,255
1160 DATA 141,1,208,141,226,6
1170 DATA 173,225,6,24,109,227,6,205,2
30,6,176,27,205,231,6,144,22,141,225,6
,141,0,208,141,2,208,141,3,208
1180 DATA 173,244,6,240,13,189,0,141,2
44,6
1190 DATA 173,227,6,73,255,141,227,6,2
06,228,6,16,78,173,245,6,205,233,6,176
,70,173,229,6,141,228,6
1200 DATA 162,11,189,128,6,240,5,202,1
6,248,48,52
1210 DATA 165,89,157,200,6,165,88,24,1
05,120,157,188,6,144,3,254,200,6,173,2
25,6,157,164,6,56,233,40
1220 DATA 74,74,24,125,188,6,157,188,6
,144,3,254,200,6,169,1,157,128,6,141,2
35,6,238,245,6
1230 DATA 162,11,189,128,6,240,110,222
,140,6,189,140,6,16,102,189,152,6,157,
140,6,189,188,6,133,208,189,200,6
1240 DATA 133,209,189,176,6,160,0,145,
208,165,208,24,105,40,133,208,157,188,
6,144,5,230,209,254,200,6
1250 DATA 254,212,6,189,212,6,201,11,1
44,28,201,17,176,24,189,164,6,24,105,9
,205,226,6,144,13
1260 DATA 56,233,18,205,226,6,176,5,14
4,30,24,144,165,189,212,6
1270 DATA 201,20,144,8,169,1,141,224,6
,24,144,42,177,208,157,176,6
1280 DATA 169,3,145,208,24,144,27,169,
1,141,234,6,169,0,157,128,6,157,212,6,
157,176,6

```


WILDWEST
Assembly Listing
by Stan Ockers
Ace Newsletter

PAGE 1

```

10 ; WILDWEST VBI ROUTINE
20 ;
0000 30      = $0680
0680 40 DFLG = $+12 ; (1664) flags for objects active
068C 50 CMT  = $+12 ; (1676) delay to drop a line
0690 60 MAX  = $+12 ; (1688) reset delay
06A4 70 HORDRP = $+12 ; (1700) hor. pos. of objects
06B0 80 CSAV = $+12 ; (1712) retain screen char.
06BC 90 POSLO = $+12 ; (1724) lo byte object pos.
06C8 000 POSHI = $+12 ; (1736) hi byte object pos.
06D4 0110 BTCNT = $+12 ; (1748) count of dropped line
06E0 0120 QUIT = $+1 ; (1760) falling routine active
06E1 0130 HORPO1 = $+1 ; (1761) shadow hor. pos. dropper
06E2 0140 HORPO2 = $+1 ; (1762) shadow hor. pos. catcher
06E3 0150 DELTA = $+1 ; (1763) incr. in dropper pos.
06E4 0160 DRPCNT = $+1 ; (1764) delay till drop
06E5 0170 DRPSET = $+1 ; (1765) reset delay
06E6 0180 RTLIM = $+1 ; (1766) right limit dropper
06E7 0190 LTLIM = $+1 ; (1767) left limit dropper
06E8 0200 SCOR = $+1 ; (1768) keep track of catches
06E9 0210 HISCOR = $+1 ; (1769) maximum # catches
06EA 0220 SNDPLG = $+2 ; (1770) sound active flags
06EC 0230 SDCMT = $+2 ; (1772) delay till next sou.
06EE 0240 SNDPOL = $+2 ; (1774) lo byte snd string
06F0 0250 SNDPOH = $+2 ; (1776) hi byte snd string
06F2 0260 SNDINDX = $+2 ; (1778) index into snd string
06F4 0270 SWFLG = $+1 ; (1780) flag to sw. dropper dir.
06F5 0280 DRPTOT = $+1 ; (1781) total # objects dropped
-0003 0290 CHAR = $03 ; char. of dropping obj.
-0010 0300 LOLIM = $10 ; lowest dropping pos.
-00D0 0310 INDLO = $D0 ; indir. pointers
-00D1 0320 INDHI = $D1
-E462 0330 XITVB = $E462 ; back to Atari VBI
-0058 0340 SAVMSC = $58 ; start of screen data
-0078 0350 OFFSET = $20 ; skip a few lines
-0270 0360 PADDLO = $0270
-D000 0370 HPOSP0 = $D000 ; PM hor. registers
-D001 0380 HPOSP1 = $D001
-D002 0390 HPOSP2 = $D002
-D003 0400 HPOSP3 = $D003
-D200 0410 AUDF1 = $D200 ; audio registers
-D201 0420 AUDC1 = $D201
-D202 0430 AUDF2 = $D202
-D203 0440 AUDC2 = $D203
-0010 0450 CATCH1 = $16 ; lowest catch line
-0012 0460 CATCH2 = $18 ; highest catch line
06F6 0470 = 0
0000 ADEA06 0480 LDA SMDPLG ; catch snd active?
0003 F039 0490 BEQ BOIT ; no
0005 CEEC06 0500 DEC SDCMT ; next sound?
0008 1034 0510 BPL DOIT ; no
000A ADEE06 0520 LDA SMDPOL ; point to snd string
000D 85D0 0530 STA INDLO
000F ADF006 0540 LDA SMDPOH
0012 85D1 0550 STA INDHI
0014 ACF206 0560 LDY SMDINDX ; position in string
0017 B1D0 0570 LDA (INDLO),Y ; get byte
0019 F015 0580 BEQ STOPSND ; if zero stop sound
001B 8D00D2 0590 STA AUDF1 ; else use as freq.
001E C8 0600 INY ; next byte
001F B1D0 0610 LDA (INDLO),Y

```

PAGE 2

```

0021 8D01D2 0620 STA AUDC1 ; is dist. and vol.
0024 C8 0630 INY
0025 B1D0 0640 LDA (INDLO),Y ; third byte is
0027 8DEC06 0650 STA SDCMT ; delay cnt.
002A C8 0660 INY
002B 8CF206 0670 STY SMDINDX ; new position
002E D00E 0680 BNE DOIT ; skip
0030 A900 0690 STOPSND LDA #0 ; shut off sound
0032 8D00D2 0700 STA AUDF1
0035 8D01D2 0710 STA AUDC1
0038 8DEA06 0720 STA SMDPLG
003B 8DF306 0730 STA SMDINDX+1 ; and reset pointer
003E D8 0740 DOIT CLD ; all math in binary
003F ADE006 0750 LDA QUIT ; do VBI?
0042 F003 0760 BEQ OVER4 ; yes
0044 4C62E4 0770 JMP XITVB
0047 AD7002 0780 OVER4 LDA PADDLO ; update catcher
004A 29FF 0790 AND #FF ; reverse paddle dir.
004C 8D01D0 0800 STA HPOSP1 ; register
004F 8DE206 0810 STA HORPO2 ; and shadow
0052 ADE106 0820 LDA HORPO1 ; update dropper
0055 18 0830 CLC
0056 6DE306 0840 ADC DELTA
0059 CDE606 0850 CMP RTLIM ; too far right
005C B018 0860 BCS SWITCH ; yes
005E CDE706 0870 CMP LTLIM ; too far left
0061 9016 0880 BCC SWITCH ; yes
0063 8DE106 0890 STA HORPO1 ; update shadow
0066 8D00D0 0900 STA HPOSP0 ; and registers
0069 8D02D0 0910 STA HPOSP2
006C 8D03D0 0920 STA HPOSP3
006F ADF406 0930 LDA SWFLG ; change dropper dir. ?
0072 F0D0 0940 BEQ DROP ; no
0074 A900 0950 LDA #0 ; reset flag
0076 8DF406 0960 STA SWFLG
0079 ADE306 0970 SWITCH LDA DELTA ; change direction
007C 49FF 0980 EOR #FF
007E 8DE306 0990 STA DELTA
0081 CEE406 1000 DROP DEC DRPCNT ; drop another?
0084 104B 1010 BPL MOVE ; no
0086 ADF506 1020 LDA DRPTOT ; have we dropped all?
0089 CDE906 1030 CMP HISCOR
008C B043 1040 BCS MOVE ; yes
008E ADE506 1050 LDA DRPSET ; reload count
0091 8DE406 1060 STA DRPCNT ; till next drop
0094 A211 1070 LDA #11 ; find a free position
0096 BD8006 1080 TRY LDA DFLG,X
0099 F005 1090 BEQ START
009B CA 1100 DEX
009C 10F8 1110 BPL TRY
009E 3031 1120 BMI MOVE ; none available
00A0 A559 1130 START LDA SAVMSC+1 ; relate starting
00A2 9DC806 1140 STA POSHI,X ; position to start
00A5 A558 1150 LDA SAVMSC ; of screen
00A7 18 1160 CLC
00A8 6978 1170 ADC #OFFSET
00AA 9DBC06 1180 STA POSLO,X
00AD 9003 1190 BCC OVER2
00AF FEC806 1200 INC POSHI,X
00B2 ADE106 1210 OVER2 LDA HORPO1 ; get dropper pos.
00B5 9DA406 1220 STA HORDRP,X ; save for coll. det.

```

```

1290 DATA 238,232,6,173,232,6,205,233,
6,176,209,202,16,196
1300 DATA 169,0,141,2,210,141,3,210,16
2,11,189,128,6,208,6,202,16,248,76,98,
228
1310 DATA 165,20,41,1,141,2,210,169,6,
141,3,210,24,144,235
1320 REM # insert VBI #
1330 RESTORE 1350:FOR J=1536 TO 1545:R
EAD A:POKE J,A:NEXT J
1340 VBI=ADR(VBI):HI=INT(VBI/256):LO=
VBI-256*HI:POKE 1538,LO:POKE 1540,HI:R
ETURN

```

```

1350 DATA 104,160,0,162,0,169,7,76,92,
228
1360 REM # init. page 6 values #
1370 FOR J=1664 TO 1675:POKE J,0:NEXT
J:FOR J=1740 TO 1759:POKE J,0:NEXT J
1380 A=ASC(DSPD*(DIF)):FOR J=1676 TO 1
699:POKE J,A:NEXT J
1390 A=ASC(DDLY*(DIF)):POKE 1764,A:POK
E 1765,A
1400 A=ASC(CNT*(DIF)):POKE 1769,A:POKE
1768,0
1410 RETURN

```



```

00B6 38      1230 SEC
00B9 E928    1240 SBC #40 ; change to screen bytes
00BB 4A      1250 LSR A
00BC 4A      1260 LSR A
00BD 18      1270 CLC
00BE 7DBC06  1280 ADC POSLO,X ; and add it in
00C1 9DBC06  1290 STA POSLO,X
00C4 9003    1300 BCC OVER3
00C6 FEC806  1310 INC POSHI,X
00C9 A901    1320 OVER3 LDA #01 ; activate drop
00CB 9D8006  1330 STA DFLG,X
00CE EEF506  1340 INC DRPTOT
00D1 A208    1350 MOVE LDX #11 ; six objects
00D3 BD8006  1360 LOOP LDA DFLG,X ; this one dropping?
00D6 F06E    1370 BEQ NEXT1 ; no
00D8 DE8C06  1380 DEC CNT,X ; down again?
00DB BD8C06  1390 LDA CNT,X
00DE 1066    1400 BPL NEXT1 ; no
00E0 BD9806  1410 LDA MAX,X ; reset drop counter
00E3 9D8C06  1420 STA CNT,X ; for next down
00E6 BDBC06  1430 LDA POSLO,X ; fill indirect pointer
00E9 85D0    1440 STA INDLO ; with position
00EB BDC806  1450 LDA POSHI,X
00EE 85D1    1460 STA INDHI
00F0 BDB006  1470 LDA CSAV,X ; restore background
00F3 A000    1480 LDY #0
00F5 91D0    1490 STA (INDLO),Y
00F7 A5D0    1500 LDA INDLO ; down a line
00F9 18      1510 CLC
00FA 6928    1520 ADC #40
00FC 85D0    1530 STA INDLO ; for indirect
00FE 9DBC06  1540 STA POSLO,X ; and new position
0101 9005    1550 BCC OVER1 ; no page crossing
0103 E6D1    1560 INC INDHI ; page crossed
0105 FEC806  1570 INC POSHI,X
0108 FED406  1580 OVER1 INC BTCNT,X ; check for a catch
010B BDD406  1590 LDA BTCNT,X
010E C910    1600 CMP #CATCH1
0110 901C    1610 BCC BOTT ; not to catch pos.
0112 C912    1620 CMP #CATCH2
0114 B018    1630 BCS BOTT ; past catch position
0116 BDA406  1640 LDA HORDRP,X ; recall horiz. pos.
0119 18      1650 CLC
011A 6908    1660 ADC #08 ; but go a bit right
011C CDE206  1670 CMP HORPO2 ; catcher further right?
011F 900D    1680 BCC BOTT ; yes
0121 38      1690 SEC
0122 E916    1700 SBC #16 ; now a little left
0124 CDE206  1710 CMP HORPO2 ; catcher further left?
0127 B005    1720 BCS BOTT ; yes
0129 901E    1730 BCC CAUGHT ; within limits, a catch
012B 18      1740 LOOP1 CLC ; just to keep
012C 99A5    1750 BCC LOOP ; everything relative
012E BDD406  1760 BOTT LDA BTCNT,X ; reached bottom?
0131 C910    1770 CMP #LOLIM
0133 9008    1780 BCC FILL ; no
0135 A901    1790 NOMORE LDA #01 ; yes stop everything
0137 8DE006  1800 STA QUIT
013A 18      1810 CLC
013B 902A    1820 BCC OUT
013D B1D0    1830 FILL LDA (INDLO),Y ; get screen character

```



```

013F 9DB006  1840 STA CSAV,X ; and save it
0142 A903    1850 LDA #CHAR ; put object
0144 91D0    1860 STA (INDLO),Y ; on screen
0146 18      1870 NEXT1 CLC
0147 9018    1880 BCC NEXT
0149 A901    1890 CAUGHT LDA #1 ; activate sound
014B 8DEB06  1900 STA SMDFLG+1
014E A900    1910 LDA #0 ; reset
0150 9D8006  1920 STA DFLG,X ; flags
0153 9DD406  1930 STA BTCNT,X ; and bottom count
0156 9DB006  1940 STA CSAV,X ; and saved char.
0159 EEE806  1950 INC SCOR ; check for all caught
015C ADE806  1960 LDA SCOR
015F CDE906  1970 CMP HISCOR
0162 B0D1    1980 BCS NOMORE ; yes stop everything
0164 CA      1990 NEXT DEX ; more objects?
0165 10C4    2000 BPL LOOP1 ; yes
0167 A900    2010 OUT LDA #0 ; shut off sound
0169 8D02D2  2020 STA AUDF2
016C 8D03D2  2030 STA AUDC2
016F A208    2040 LDX #08 ; search for active object
0171 BD8006  2050 LOOP3 LDA DFLG,X
0174 D006    2060 BNE FIZZ ; found one
0176 CA      2070 NEXTPO DEX
0177 10F8    2080 BPL LOOP3
0179 4C62E4  2090 JMP AITVB
017C A514    2100 FIZZ LDA #14 ; random sound
017E 2901    2110 AND #1
0180 8D02D2  2120 STA AUDF2 ; sound on
0183 A906    2130 LDA #6
0185 8D03D2  2140 STA AUDC2
0188 18      2150 CLC
0189 90EB    2160 BCC NEXTPO

```

PAGE 5 SYMBOLS

=D201 AUDC1	=D203 AUDC2	=D200 AUDF1	=D202 AUDF2
06D4 BTCNT	=0010 CATCH1	=0012 CATCH2	0149 CAUGHT
068C CNT	06B0 CSAV	06E3 DELTA	0680 DFLG
0081 DROP	06E4 DRPCNT	06E5 DRPSET	06F5 DRPTOT
017C FIZZ	06E9 HISCOR	06A4 HORDRP	06E1 HORPO1
=D000 HPOSP0	=D001 HPOSP1	=D002 HPOSP2	=D003 HPOSP3
=00D0 INDLO	=0010 LOLIM	00D3 LOOP	012B LOOP1
06E7 LTLIM	0698 MAX	00D1 MOVE	0164 NEXT
0135 NOMORE	0176 NEXTPO	=0078 OFFSET	0167 OUT
00B2 OVER2	00C9 OVER3	0047 OVER4	=0270 PADDLO
068C POSLO	06E0 QUIT	06E6 RTLIM	=0058 SAVMSC
06EC SMDCNT	06EA SMDFLG	06F2 SMDINDX	06F0 SMDPOH
00A0 START	0030 STOPSMD	06F4 SWFLG	0079 SWITCH
=E462 XITVB			

→ Doggies

```

2530 DATA 496,240,240,240,226,194,226,
244,248
2540 DATA 504,240,240,240,224,192,224,
240,248
2550 DATA -1
2559 REM *** Return to user ***
2560 POSITION 9,22: "Press START to
begin":POKE 53279,8
2570 FOR I=1 TO 20:IF PEEK(53279)<>6 T
HEN NEXT I:POKE 755,2-PEEK(755):GOTO 2
570
2580 POP :GRAPHICS 18:POKE 16,64:POKE
53774,119:POKE 756,START:POKE 559,46:P
OKE 53277,3
2590 POKE 708,14:POKE 709,184:POKE 710
,20:POKE 711,136:A=USR(1536):RETURN

```



FALLING

Listing one
by Stan Ockers

PAGE 1

```

10 ; SKY IS FALLING ROUTINE
20 ;
0000 30      = $0680
0680 40 DFLG = $+6
0686 50 CMT  = $+6
068C 60 MAX  = $+6
0692 70 STLO = $+6
0698 80 STHI = $+6
069E 90 POSLO = $+6
06A4 0100 POSHI = $+6
06AA 0110 BTCNT = $+6
06B0 0120 CSAV = $+6
      0130 CHAR = $03
      0140 LOLIM = $10
      0150 INDLO = $D0
      0160 INDHI = $D1
      0170 XITVB = $E462
0180 ;
06B6 0190      = $00
0000 D8      0200 CLD      ; add in binary
0001 A205     0210 LDX #5   ; six objects
0003 BD8006   0220 LOOP LDA DFLG,X ; this one dropping?
0006 F05D     0230 BEQ NEXT ; no
0008 DE8606   0240 DEC CMT,X ; down again?
0008 BD8606   0250 LDA CMT,X
000E 1055     0260 BPL NEXT ; no
0010 BD8C06   0270 LDA MAX,X ; reset drop counter
0013 9D8606   0280 STA CMT,X ; for next down
0016 BD9E06   0290 LDA POSLO,X ; fill indirect pointer
0019 85D0     0300 STA INDLO ; with position
001B BDA406   0310 LDA POSHI,X
001E 85D1     0320 STA INDHI
0020 BDB006   0330 LDA CSAV,X ; restore background
0023 A000     0340 LDY #0
0025 91D0     0350 STA (INDLO),Y
0027 A5D0     0360 LDA INDLO ; down a line
0029 18      0370 CLC
002A 6928     0380 ADC #40
002C 85D0     0390 STA INDLO ; for indirect
002E 9D9E06   0400 STA POSLO,X ; and new position
0031 9005     0410 BCC OVER1 ; no page crossing
0033 E6D1     0420 INC INDHI ; page crossed
0035 FEA406   0430 INC POSHI,X
0038 FEAA06   0440 OVER1 INC BTCNT,X ; reached bottom?
003E C910     0450 LDA BTCNT,X
0040 901A     0460 CMP $LOLIM
0042 A900     0470 BCC FILL ; no
0044 9D8006   0480 LDA #0 ; reset
0047 9DAA06   0490 STA DFLG,X ; flags
004A 9DB006   0500 STA BTCNT,X ; and bottom count
004D BD9206   0510 STA CSAV,X ; and saved char.
0050 9D9E06   0520 LDA STLO,X ; reset position
0053 BD9806   0530 STA POSLO,X ; to starting pos.
0056 9DA406   0540 LDA STHI,X
0059 18      0550 STA POSHI,X
005A 9009     0560 CLC ; don't put char.
005C B1D0     0570 BCC NEXT ; on screen
005E 9DB006   0580 FILL LDA (INDLO),Y ; get screen character
0061 A903     0590 STA CSAV,X ; and save it
0063 91D0     0600 LDA $CHAR ; put object
      0610 STA (INDLO),Y ; on screen

```

PAGE 2

```

0065 CA      0620 NEXT DEX ; more objects?
0066 109B     0630 BPL LOOP ; yes
0068 4C62E4   0640 JMP XITVB ; back to VBI routine

```

```

1 REM *****
2 REM **ATARI COMPUTER ENTHUSIASTS**
3 REM *****NEWSLETTER*****
4 REM **      3662 Vine Maple DR      **
5 REM **      EUGENE OR 97405      **
6 REM **      April 83              **
7 REM **      $10 yr                 **
8 REM *****

```

```

0686 CNT
=00D0 INDLO
0038 OVER1
=E462 XITVB

```

```

06B0 CSAV
=0010 LOLIM
06A4 POSHI

```

```

0680 DFLG
0003 LOOP
069E POSLO

```

```

BY STAN OCKERS
9 REM *** DEMO OF FALLING ROUTINE ***
10 GOSUB 8000:GOSUB 8200:GOSUB 8300
20 FOR J=1 TO 500:NEXT J
30 R=RND(0)*6:POKE 1664+INT(R),1
40 GOTO 20
7999 REM *** CREATE VBI STRING ***
8000 DIM VBI$(110):RESTORE 8010:FOR J=
1 TO 110:READ A:VBI$(J,J)=CHR$(A):NEXT
J:RETURN
8010 DATA 216,162,5,189,128,6,240,93,2
22,134,6,189,134,6,16,85,189,140,6,157
,134,6,189,158,6,133,208,189,164,6
8020 DATA 133,209,189,176,6,160,0,145,
208,165,208,24,105,40,133,208,157,158,
6,144,5,230,209,254,164,6
8030 DATA 254,170,6,189,170,5,201,19,1
44,26,169,0,157,128,6,157,170,6,157,17
6,6,189,146,6
8040 DATA 157,158,6,189,152,6,157,164,
6,24,144,9,177,208,157,176,6,169,3,145
,208,202,16,155,76,98,228
8199 REM *** INSERT VBI ***
8200 RESTORE 8230:FOR J=1536 TO 1545:R
EAD A:POKE J,A:NEXT J
8210 VBI=ADR(VBI$):HI=INT(VBI/256):LO=
VBI-256*HI:POKE 1538,LO:POKE 1540,HI:A
=USR(1536):RETURN
8230 DATA 104,160,0,162,0,169,7,76,92,
228
8299 REM *** IMIT. PAGE 6 VALUES ***
8300 RESTORE 8310:FOR J=1664 TO 1681:R
EAD A:POKE J,A:NEXT J
8310 DATA 0,0,0,0,0,0,0,0,0,0,10,8
,16,14,12,10
8312 FOR J=1719 TO 1725:POKE J,0:NEXT
J
8320 SCR=PEEK(88)+256*PEEK(89):TOPLN=5
CR+120:BOTLN=SCR+800
8322 T1=TOPLN*8:HI=INT(T1/256):LO=T1-2
56*HI:POKE 1682,LO:POKE 1694,LO:POKE 1
688,HI:POKE 1700,HI
8324 T2=TOPLN*16:HI=INT(T2/256):LO=T2-
256*HI:POKE 1683,LO:POKE 1695,LO:POKE
1689,HI:POKE 1701,HI
8326 T3=TOPLN*24:HI=INT(T3/256):LO=T3-
256*HI:POKE 1684,LO:POKE 1696,LO:POKE
1690,HI:POKE 1702,HI
8328 T4=TOPLN*32:HI=INT(T4/256):LO=T4-
256*HI:POKE 1685,LO:POKE 1697,LO:POKE
1691,HI:POKE 1703,HI
8330 T5=TOPLN*40:HI=INT(T5/256):LO=T5-
256*HI:POKE 1686,LO:POKE 1698,LO:POKE
1692,HI:POKE 1704,HI
8332 T6=TOPLN*48:HI=INT(T6/256):LO=T6-
256*HI:POKE 1687,LO:POKE 1699,LO:POKE
1693,HI:POKE 1705,HI
8340 FOR J=1706 TO 1717:POKE J,0:NEXT
J
8350 RETURN

```

PAGE 3
SYMBOLS

```

06AA BTCNT      =0003 CHAR
005C FILL      =00D1 INDHI
068C MAX        0065 NEXT
0698 STHI       0692 STLO

```



OCKERS

WILD WEST

Wildwest uses an expanded version of the 'Sky is Falling' routine. Sound is included in the VBI routine along with dropping and catching routines.

Timing is most critical in this program and you may wish to change some of the characteristics. Most of the timing is controlled by the data in three strings. Each byte in the string represents one difficulty level. DSPD\$ (lines 1030,1040) bytes control the speed at which the dynamite falls. Lower numbers mean faster speeds. DDLY\$ (lines 1050,1060) holds bytes which determine the delay time until another stick is dropped. CNT\$ (1070,1080) determines the total number of sticks dropped in any one group. The number used for comparison with the random number in line 250 determines how often Dan switches direction. Increase the value to make him change direction more frequently. Experiment with some of the values and try to come up with the most challenging combinations.

You may wonder why collision registers were not used to detect collisions. I tried, but it seems these registers are updated during Atari's portion of the VBI routine. Since I needed to know which object had collided and many might be moved during one VBI pass, there seemed no way to use collision registers for detection.

The program works best with paddles but if you don't have them, make the following changes to be able to use a joystick:

- (1) replace line 1150 with: 1150 Data 162,0,173,120,2,41,4,208,2,162,252,173,120,2,41,8,208,2,162,3,138,109,226,6
- (2) change the 398's in line 1110 to 417 (two places)
- (3) change 'paddle zero' in line 640 to 'joystick 0'
- (4) change PTRIG(0) in line 230 to STRIG(0).

I find the joystick very difficult to use. You might find the action more to your liking by replacing the first 5 bytes of 1150 above with 234's.

—Stan Ockers

Machine Language Programming #5

Sky is Falling Routine

Many games involve falling objects. Player missiles will suffice for a few objects but if many are involved another method is necessary. We can use the techniques we have been studying to write what I call a 'Sky is Falling' routine. Listing 1 is such a program.

The idea is to use a character mode (mode 0 in this case) and move objects (characters) by removing them from one line and placing them on the next line down. The program is set up for six such objects but could easily be changed for more. Characters are POKEd directly into screen data memory. A major part of the program involves calculating where the characters are located.

Since there are 40 characters per line, the new position of an object is 50 bytes from the previous position. Object positions are kept in a table of high and low bytes (POSLO & POSHI and following). X acts as an index to choose which pair of bytes we want. The positions are initially set (from Basic) to the top of the screen. Other bytes (STLO,STHI) hold those positions so they can be reset when the bottom-most positions are reached.

The indirect indexed mode is used to point to the position required. After adding 40 to the location, the character at the new position is saved. This character will be restored when the object moves. This allows the falling objects to pass in front of a background without creating 'holes'. Notice the index used is actually zero, for we are interested mainly in the indirect nature of the mode.

Another technique to notice in the program is the use of flags. A set of flags keep track of whether or not an object is falling. Changing the flag from zero to nonzero initiates the dropping of the object.

If we try using the USR function to handle movement of objects we will find they can't be moved fast enough. Instead the routine is set up to operate during the Vertical Blank Interrupt. The VBI happens every 1/60th of a second when the beam creating the TV picture retraces from the bottom to the top of the screen. The routine itself is put into a string (VBI\$) and the address of the string is passed to a routine which inserts it into the vertical blank interrupt sequence (see the Basic listing).

Now the positions of objects will be updated sixty times a second. This may be too fast! To slow things down, counters are used to regulate the speed (CNT). These are reset (with MAX) after counting through zero. Movement is performed only after each complete countdown.

Another counter keeps track of how many lines each object has dropped (BTCNT). When it reaches a certain value (LOLIM) the character is not printed. The dropping is deactivated and the screen position bytes are reset to the top of the screen.

We don't use 'Hexpoke' this time. Instead we change the code to decimal and use Basic to handle it. The Basic program listed demonstrates the falling routine. Graphics 0 is used while in an actual program mode 4 could be substituted for more color. An actual program might also have the object dropping from positions depending upon another object (player missile). In such a case additional code is required in the VBI routine. Catching of dropped objects or collision detection also requires additional code.

Careful study of this routine should help clarify techniques we have introduced in past installments. The concepts of addition and page crossing are also included though not previously described. These we will save for a future installment.

—Stan Ockers

CALL TO ASSEMBLY

from S*P*A*C*E by Tom Newman

MICRO-PAINTER Part 3:

"The uncompactor"

In part two of this article we created a disk file containing the compacted data of our MICRO-PAINTER picture. This article will show you how to load the compacted file back into RAM, using a portion of the program from the first part of the article. We will also have to create a mode 'E' display list so we can view our picture as the program unfolds it. Again we will use a portion of the first program. Keep in mind the screen memory and mode 'E' display list is a necessary overhead in displaying pictures created by MICRO-PAINTER.

One of the possible uses of this program is to store several pictures in RAM, and then unfold them one at a time into a common screen RAM area. This can be useful as a tutorial, where a series of complex, rapidly available pictures are desirable to clarify a point. My sons, of course, have a different interest in the advantages of the compactor program. It lets them get all four of the DONKEY KONG screens into the computer's memory at the same time, which furthers their hopes I will put some action into the pictures they have so artfully created.

In using the MICRO-PAINTER pictures in a game, I suggest you start by choosing one with a large static playfield. Then use players for the moveable objects. There are collision detection registers built into the Atari hardware, which can detect contact between the players and the playfield, or each other. The use of these registers is outlined in Atari's hardware manual (CO16555). To get an idea of what can be done with bit mapped graphics, take a look at BRODERBUND'S CHOPPLIFTER for an outstanding example. This is the type of thing which can be accomplished with these pictures, not with as much resolution, but the same type of action and much more control over the colors.

At this point I will leave the possibilities up to your imagination and get down to an explanation of the un-compaction program.

Lines 130 to 230 contain the equate table. If you are using the Atari ASSEMBLER/EDITOR cartridge use zero page locations \$B0 to \$B8 instead of \$F0 to \$F8. Each assembler (hopefully) will have some zero page locations set aside for your use. The exact locations vary from one assembler to another, so be certain the locations you use are free, or you may discover the gates to never never land.

Lines 270 to 380 initialize the program variables, like where to put the uncompacted data and where to get it.

Lines 420 to 800 contain the program body. First we get a byte from the data table we created in the last article. The table is indexed by the 'Y' register to first point to the data byte and then the repeats byte. The actual movement through the table however is accomplished by altering the zero-page pointer 'TABLE', and then resetting the 'Y' register to zero upon each access to the table. I also have used the 'Y' index register to keep track of which column is being put to the screen. The column number is saved in a temporary location called 'SAVE' while the data table is being accessed.

In the mode 'E' standard width screen there are forty bytes per line. In order to put one byte directly under another in the screen RAM to create the columns we must add forty (HEX 28) to the previous address. This is accomplished in lines 580 to 640. If the end of the column has been reached then the buffer is restored to its starting value, and the 'Y' register incremented. When all of the columns are transferred then the program is exited.

Now it is time to take a look at your picture. You must have the mode 'E' display list in memory, and point the display list pointer at the new list. Use lines 530 through 980 of the program on page 9 of the November newsletter to accomplish this. Assemble the program to disk (.TF "D:YOURFILE"), so you may use it later. Now assemble the program in this article into RAM. (DON'T FORGET TO SAVE IT). BLOAD the compacted file you created from the last article, and the display list setup mentioned above. To execute the program type MON then 5F00G (THE SCREEN WILL GO BLANK), listen for the bell! Carefully type 5E00G and your picture should appear on the screen.

Good luck.
—T.E.N.



TOOLS OF THE TRADE

(By permission from **Programmer's Journal**, Box 5070, Eugene, OR 97405; \$24/6 issues; ed. Greg Estes; Resource Newsletter for IBM PC Programmers.)

[This article focusses on Microsoft BASIC for the IBM PC, but I feel the ideas discussed are equally appropriate for programmers in ATARI BASIC. —JB, ed.]

Programming in any language goes through a process of evolution. The standard tools of BASIC we will cover in this column are very useful in themselves. The real concern is establishing awareness how our software changes and improves with age and attention and, most of all, by contact with other good software writers.

Our first focus is often just "getting the job done", no matter what the cost in code size or structure. As our skills and understanding of the language improve we are able to reduce the size of the program while still keeping or increasing the clarity of its function. We will quickly cover the evolution of some single line BASIC tools before moving on to larger ones like input routines.

When a YES or NO question is asked a user we must determine which response is made and execute the appropriate choice. Initially we code it like this:

```
100 INPUT "Do you want to try again (Y/N)";A$
110 IF A$ = "Y" THEN GOSUB 500
120 IF A$ = "Y" THEN GOSUB 500
130 IF A$ = "N" THEN GOSUB 600
140 IF A$ = "N" THEN GOSUB 600
150 GOTO 100
```

Lots of code. Soon we figure out we can shorten it slightly by checking each answer in one line:

```
110 IF A$ = "Y" OR A$ = "y" THEN GOSUB 500
120 IF A$ = "N" OR A$ = "n" THEN GOSUB 600
130 GOTO 100
```

This is better, and for a yes or no question certainly suffices. But what if we had a menu with 5 choices and the options were A, B, C, D, or E?

Microsoft gives us a standard tool for this, the INSTR function. It searches a string for a character and returns the number of where it found it, or a zero if it failed. Our yes/no question can now be coded like this:

```
110 NUMBER = INSTR(1,"YyNn",A$)
120 ON NUMBER GOSUB 500,500,600,600
130 GOTO 100
```

As time goes on we notice the INSTR function itself gives us the number we need, so why bother with the NUMBER variable at all. This enables us to handle a list of menu options (say A-E) in only one line:

```
110 ON INSTR(1,"aAbBcCdDeE",A$) GOSUB 500,500,600,600,700,700,800,800,900,900
120 GOTO 100
```

Every function in BASIC has numerous uses. As our experience and creativity in a language evolve we discover new uses for old functions. Many such ideas are passed from programmer to programmer with improvements and new tricks added along the way.

UPPERCASE AND LOWERCASE: There are many times we want to convert pieces of data to uppercase from lowercase. For instance, in a mailing list program the 2 letter state field should always be forced to uppercase even if the user forgets. Often the first letter of first or last names are also forced to uppercase, especially when building index files or searching for a matching name you normally want to compare fields both of which we uppercase to catch every occurrence. "Johnson" should match "JOHNSON" unless otherwise indicated. In order to do this we check to see if a letter is within the lowercase a-z range, and if it is just make it uppercase by subtracting 32 from its ASC value. Look at Appendix G in the BASIC manual. We have several functions we can put together to do this, such as the ASC and CHR\$ functions. For example:

```
200 C$ = "b"
210 C$ = CHR$(ASC(C$)-32)
```

The result "b" is changed to "B". With larger variables we need to do it in a loop. The LEN function gives us the number of characters in our variable. We are going to put the letters one at a time into a C\$ field to check them. The MID\$ function allows us to extract pieces of words. Let's assume we have a state code called ST\$ and we want to convert the whole thing to uppercase.

```
200 FOR I = 1 TO LEN(ST$):REM length of state
210 C$ = MID$(ST$,I,1):REM get it
220 IF C$ = "" THEN GOTO 250:REM possible error
230 IF C$ = "a" AND C$ = "z" THEN C$ = CHR$(ASC(C$)-32)
240 MID$(ST$,I,1) = C$:REM put it back
250 NEXT I
```

IBM PC BASIC from Microsoft ALMOST gave us a complete tool to do this with the Define Function capability. There were a few grumblings from the programming community when the DEF FN (defining functions) was only allowed a one equation length. The problem is apparent when we want to force a field (our STATE field) to uppercase. We want to check and see if a letter was already capitalized, and then capitalize it all in one fell swoop. This requires 2 statements, and the DEF FN only allows us 1. So we do the best we can. Defining a function we get:

```
20 DEF FNCAP$(A$) = CHR$(ASC(C$)-32)
```

This method is ideal for numeric equations but can often be useful with strings. Remember to end the definition with a \$ if it is a string definition and put the code physically ahead of the function. NOTE: I spent 2 hours one Saturday morning wondering why one program line kept getting a SYNTAX error. I was writing a mailing list program and I had changed a variable name from FIRSTNAME\$ to FNAME\$. The only variable name you cannot use other than reserved words is a variable beginning with FN because Microsoft BASIC considers it to be a previously defined function. So do not ever use a variable beginning with FN unless it is a special user defined function. Line 230 of the above statement can be rewritten using the defined function.

```
230 IF C$ = "a" AND C$ = "z" THEN C$ = FNCAP$(C$)
```

DIVIDING BY ZERO: This is a perpetual problem. We should always check any number to be used for division to see if it is equal to zero.

```
200 IF X = 0 THEN GOTO 220
210 AMT = AMT/X
220 ...
```

You can abbreviate this by just saying "IF X" which is the equivalent of saying "IF X

0". The "IF" here is going to see if the variable or equation is TRUE or FALSE. TRUE means a non-zero and FALSE means a zero value. Rewriting we can say:

```
200 IF X THEN AMT = AMT/X ELSE AMT = 0 ..... and so forth
```

MONTH/DAY/YEAR: Dealing with dates is as much fun as giving your cat a bath. One approach is to convert dates from our MM/DD/YY calendar (Gregorian) to a Julian date or absolute number. A chunk of code which takes both time and space. Since a lot of information our computer handles is dated, and it is something we have to SORT on, this situation is repeatedly staring us in the face. A simpler approach to solve much of the dilemma is to move the Year to the front of the date so you have a YEAR/MONTH/DAY field. So whether the field is saved as a string or a number it still sorts out correctly by date. (Don't forget the zeros.)

```
12/01/82 is stored as "821201"
01/02/83 is stored as "830102"
```

HOUSEKEEPING: Often you can spot a good piece of code written by an experienced programmer by looking at the level of his housekeeping or initialization at the beginning of the program. We will mention a few and explain their usage.

```
20 DEFINT A-Z : TRUE = -1 : FALSE = 0 :CLS :SCREEN 0,0,0
30 ESC$ = CHR$(27) : RTN$ = CHR$(13) : BS$ = CHR$(8)
40 CTLB$ = CHR$(3) : FOR I = 1 TO 10 : KEY I,"": NEXT I
```

The DEF INT makes all the numeric variables you use of type integer unless otherwise stated. This saves a lot of space should you use many numbers and also speeds things up. The next steps are to set up a few variables to make our program more readable. You can set a variable to TRUE or FALSE at one place in the program and then check it later.

```
100 OVERDUE = TRUE
110 NEWSTUDENT = FALSE
```

```
2005 IF OVERDUE = TRUE THEN ....
2510 IF NEWSTUDENT = FALSE THEN ...
(or you can say) 2505 IF OVERDUE THEN ...
2510 IF NOT NEWSTUDENT THEN ...
```

This really helps when you are trying to narrow down program bugs and find out what you are doing. Along with this we are going to regularly check the user input to see if they typed a Backspace (BS\$) or a Return (RTN\$) or a Control Break (CTLB\$). We also set the function keys to nothing so they won't cause us trouble.

INPUT ROUTINE: We are going to spend the remainder of this column examining the evolution of an input routine, our next issue will cover it in full. An input routine is a MAJOR building block of decent software. It is the one foundation upon which decent micro computer software is built. A user spends most of the time answering questions and inputting data based upon your input routine. A poor input routine will make an otherwise excellent piece of software unusable. It is a routine in every program involving input, whether numeric or string. We will begin at the beginning.


```
100 INPUT "How old are you ?";A
```

This works...sort of. But we want to produce some quality programs. The drawbacks of this input technique are well known. The users can type 100 numbers if they want to, or type in a letter which causes a "Redo from start" and destroys our screen. We have very little control over the cursor.

```
100 INPUT "How old are you";A$
110 A = VAL(A$)
120 IF A < 1 OR A > 100 THEN GOTO 100
```

One small step for mankind here. At least we got rid of the "Redo from start" message. The A = VAL(A\$) produces the number again from the string. Also we looped around if the number was not within the range. Still the user can hit the ESCAPE key and erase the screen line. Not good at all. What we want is a statement trapping very piece of input and allowing us to keep or throw away what we want. This function is INKEY\$. This little gem gives us every keystroke from the keyboard one at a time. The computer is in fact perpetually reading the keyboard. (If you doubt this then try using INKEY\$ and set the TRON key - F7.) All we have to do is build a first class routine around it and we are well on our way to producing "friendly" and easy to use software.

Let's improve our input by locating the screen line and then using INKEY\$. We must accept the keystroke from INKEY\$ to a string variable, let's call it INK\$.

```
50 LOCATE 10,10: PRINT "Do you want to try again ? (Y/N)";
60 GOSUB 1000
70 END
```

```
1000 REM... INPUT ROUTINE
1010 INK$ = INKEY$ : IF INK$ = "" THEN GOTO 1010
1020 IF INK$ = ... sort out what you want, and return if done
```

```
1200 GOTO 1010... if more than 1 char desired, loop
```

Line 1010 sets up the endless loop until a key is hit. So once we have something, what will we do with it? We have trapped everything from the ESC key to the Control Break key to the Backspace key. The decisions you make here determine how friendly USERS consider your software. So we will throw out a few keys and keep some others to make a minimal routine understandable before considering a full length version next time. It all boils down to one concept: Keep collecting the keystrokes from the user until he hits the return key, which means they are all done. If a keystroke is good, save it. If it is not, ignore it. Fleshing out our routine a little we will add a WORD\$ variable. This is the place where the keystrokes are saved.

```
10 REM.....HOUSEKEEPING
20 DEFINT A-Z : TRUE = -1 : FALSE = 0 :CLS
30 ESC$ = CHR$(27) : RTN$ = CHR$(13) : BS$ = CHR$(8)
40 CTLB$ = CHR$(3)
50 REM.....USER INPUT
60 LOCATE 10,10: PRINT "What is your Name ?";
70 GOSUB 1000
80 PRINT WORD$
90 END
1000 REM... INPUT ROUTINE
1010 INK$ = INKEY$ : IF INK$ = "" THEN GOTO 1010
1020 IF INK$ = RTN$ THEN RETURN
1030 WORD$ = WORD$ + INK$
1200 GOTO 1010... if more than 1 char desired, loop
```

Type this short program in and watch what happens. When you hit keys nothing shows up. INKEY\$ sends nothing to the screen. This is good for password protection and the like but not for user input. They need to see what they are doing. Once you hit the RETURN key the subroutine ends and you do eventually print the WORD\$. Here we begin adding, line by line, what we need to the skillfully flesh out our input routine. Let's add some printing. (Don't forget the semi-colon.)

```
1080 PRINT INK$;
```

Seems to work. The ESC and Backspace keys look funny. We will stop them later. But how about the Control Break key? We have stopped the break exit unless the user is really fast. How about one more addition before we close shop for the day. Make the user stop at 10 characters. If the 10th letter is NOT a return we will beep and try again.

```
1030 IF LEN(WORD$) = 10 THEN BEEP :GOTO 1010
      (Delete line 80 too.)
```

All together it looks like this:

```
1000 REM.....INPUT ROUTINE
1010 INK$ = INKEY$ : IF INK$ = "" THEN GOTO 1010
1020 IF INK$ = RTN$ THEN RETURN
1030 IF LEN(WORD$) = 10 THEN BEEP :GOTO 1010
1080 PRINT INK$;
1090 WORD$ = WORD$ + INK$
1200 GOTO 1010... if more than 1 char desired, loop
```

This is only the start of a good input routine. We should ignore other keys and print a prompt line allowing the user to backspace and correct a mistake. Using the MID\$ function we can add and subtract characters from our WORD\$. Next issue we will put a complete routine together with all the bells and whistles you can plug into any program and have first class, bug free input. Stay tuned...

—Greg Estes

ERACE REVIEW

LEARNING, DISABLED STUDENTS AND COMPUTERS:

A Teacher's Guide Book

(by Merianne Metzger, David Oullette, and Joan Thorman. Published by The International Council for Computers in Education [ICCE], 135 Education, University of Oregon, Eugene, OR 97403).

Our group is delighted to be able to review a book which has been co-authored by an ACE member, Merianne Metzger. This brief 48 page booklet is one of several published by ICCE at the University of Oregon. ICCE also publishes "The Computing Teacher", an educational journal which focuses on the practical application of computers within a variety of educational settings.

The booklet presents an overview of Learning Disabilities (LD) and the rationale for the use of computers with LD students. Included is information about software and hardware. The guide book also contains references for magazines, books, organizations, networks, research, and other bibliographies which can serve as additional resources for the educator.

Although this booklet is directed at LD computer applications, much of the information and several references are appropriate for other educational computer use.

We feel the reference section is of great value to computing educators, even though computer using groups are omitted as an educational resource. If the booklet has a weakness, it may be the failure to address the following critical issues:

1. The need for research validated software;
2. Testing techniques which match learner needs with appropriate computer assisted instruction (CAI);
3. The role of the computer as an aid in mainstreaming the handicapped.

This booklet is a valuable first step for educators entering the world of computers. Terminology, common concerns and hardware considerations are explained clearly and concisely. It also serves as a good introduction to the series of books and materials published by ICCE which were designed to assist computer using educators.

—Alice Miles Erickson (ERACE)

Saundra Hopkins (Applied Computer Enterprises and Services.

COURSEWARE IN THE CLASSROOM:

Selecting, Organizing and Using Educational Software

(Ann Lathrop and Bobby Goodson, Addison-Wesley, 1983, \$10. Softcover, 187 pages.

Ann Lathrop and Bobby Goodson have prepared a much needed reference book for teachers interested in using microcomputer software. **Courseware in the Classroom** is a practical guide to integrating educational software into the existing curriculum and managing it once obtained.

The organization of the book is very straightforward and logical. In fact, the design allows the information to be easily expanded into a series of inservice workshops adequately covering all concerns teachers and administrators might have about microcomputer courseware. The material is completely understandable and useful.

Lathrop and Goodson, who encourage the gradual introduction of micros into the school, skip the usual introduction to microcomputers and begin with suggestions on how to handle scheduling, movement, and classroom setup, and how to deliver, demonstrate and otherwise manage courseware shared by many teachers and students. In a manner useful to novices to computers in education, the authors next give an overview of software applications in all areas of the curriculum. The second section of the book, consisting of six chapters, deals specifically with several such applications, namely reinforcement and remediation, tutorials, simulations and demonstrations, approaches to problem solving, program development aids (such as authoring languages), and tools for teachers (including gradebook, curriculum management, test and exercise generating, and statistics programs). Each chapter is illustrated with screen displays of popular programs which typify the particular application in various subject areas. The authors define basic qualities each type of courseware should have.

The third section of **Courseware in the Classroom** deals with the important role of courseware evaluation. After referring the reader to Appendix C for sources of reviews, Lathrop and Goodson develop a lengthy list of evaluation criteria. By covering almost everything included on well-known evaluation forms, they have made Chapter 10 especially functional to anyone who has the responsibility of evaluating courseware. This information is useful for the identification of the best forms to use, or for the development of a school's own evaluation instrument, as well as for better understanding of published reviews. To further aid the evaluator, the next chapter contains completed evaluations done on three different forms and refers the reader to Appendix B for other copiable forms. The final chapter in this section describes the complete evaluation process from start to finish.

What happens to your software after it has been purchased? Section IV offers suggestions for organizing and managing the collection in a courseware library. Also included are guidelines for a policies and procedures manual which will govern the processing, circulation and protection of the collection.

The final section of the book is a courseware directory, complete with an order form for an annual supplement. One hundred and twenty programs designed for use on Apple, Atari, Commodore PET, and TRS-80 systems are described. Most are accompanied with references to reviews recommending purchase of the program.

The appendices include information on copyright regulations, a listing of publishers of cited courseware, and a copy of "Policies and Procedures for Selection of Instructional Materials" adopted by the American Association of School Librarians.

Courseware in the Classroom is an excellent text for an introductory course or workshop in computers in education. There is no difficult theory or terminology to wade through, just practical, understandable suggestions and guidelines. The reader will finish the book with a sense he or she has gained some useful information easily implemented in any school setting.

—R. DeLoY Graham

THIRD WORLD ATARI

Welcome fellow ACERS (ACE USERS) to the international world of Atari. I'm located 12 degrees North of the equator and for all practical purposes, have fallen of the edge of the world. I'm not talking the Bahamas or Barbados here, but Grenada, a spice island 12 miles by 22 miles long.

The nearest computer shop is 1500 miles to the North. So, what you bring, is what you get. It might as well be light-years away and it's a long expensive haul to zip up to the nearest source for Atari supplies. I had to send to the U.S. to get an adaptor - going from 3 plug with ground to 2 plug without ground - they just don't exist here !!!

Mail is strange and makes the U.S. system seem like the best in the world. If anything is sent other than 1st class, you're talking 4 to 6 months delivery time. 1st class will make it from the U.S. to Grenada within a month + or - 2 weeks. Much of our mail is sent to Europe before it gets here; why, I don't know, nor does anybody else. (Could be the name Grenada???)

I needed the TEXT-WIZARD so I called some friends in the U.S. at \$15.00 a minute and they quickly ordered it. Somewhere between the U.S. and here the package got marked "Twilight Zone" and disappeared for a number of weeks.

Finally I got word a package had come to the post office and I could come down and pick it up. I gave the Post Comrad my slip and hoped I wouldn't have to pay too much duty on the package. I watched with anticipation as the man went and found my TEXT-WIZARD I had been waiting for. Without any warning, he took the package, threw it on the hard concrete, and kicked it across the floor. I was heart broken at the thought my delicate disc was probably now crushed binary soup.

He picked up the box and put it on the counter indicating I needed to sign for the package. As I was signing, he opened the package with a large knife and immediately started spraying the contents with Raid. Thousands of huge red ants appeared from the package as I dug through the box looking for what was left of my disc.

Luckily, the disc and documentation were in sealed plastic. Unluckily, the ants had eaten through the plastic. I didn't have to pay duty on the package and the post office was glad to be rid of me. I finally worked up the courage to open the sealed TEXT-WIZARD and sure enough, ants had invaded the disc as well. When I got home I let the disc sit for a few hours and the remaining ants crawled out of the disc. P.S. TEXT-WIZARD works fine!!!!!! Their product and packaging technique makes the Timex test look like a wimp fighting out of a paper bag.

Well, the TEXT-WIZARD is just great for writing all types of documents and I have found it very helpful. What I need is the updated version allowing you to use its command functions on the OKI-DATA 82-A.

I have written the people at DATASOFT and they informed me you could get an updated version BUT - they had not worked out copyright provisions with the authors. That was some time in November and I'm still waiting to hear for them... any suggestions?????

One function either not well documented or just slipped by me was the use of REPLACE. I was under the impression you had to replace each entry each time. For example, let's say I had spelled environment incorrectly throughout an article I had written. The procedure is to:

1. Press Start and the letter "S"
 2. The computer flashes on screen "SEARCH PHRASE"
 3. You type in the phrase to be searched for (environment)
 4. Press Start and the letter "R"
 5. The computer flashes on screen "REPLACE PHRASE"
 6. You type in the phrase to replace the search phrase (environment)
 7. You press Select and the letter "S"
 8. It will stop at every "environment" in the document.
 9. If you choose to replace this phrase, you press Select and the letter "R" and the phrase environment is replaced with environment.
- If you want a universal replace of environment with environment without examining each environment phrase you alter sequences 7, 8, and 9. All you have to do is: Press Select and the letter "R" holding these down until you reach the end of your document. It will find and replace each phrase you selected in 3 regardless of race, religion or creed. Of course, this information is on page 26 of the TEXT-WIZARD manual, but since it took me at least a year to find it, I thought I'd save you the trouble.

Cards, calls and letters accepted anytime...

—LINT HUTCHINSON
COUNSELING OFFICE
ST. GEORGE'S UNIVERSITY
SCHOOL OF MEDICINE
ST. GEORGE'S, GRENADA, WEST INDIES

SPELL WIZARD

(Datasoft, under \$100.00)

Here is a product which works. There are many spelling programs on the market, but this is the first one made just for the Atari computer. Not only does this program correct your spelling, but it does it in the quickest manner possible, and with as little effort as possible.

SPELL WIZARD has a dictionary of 33,000 plus words with a provision to add words it does not contain. Since SPELL WIZARD is essentially a spelling checker, a word you use may be spelled correctly, but be in the wrong tense, so the program will pass it by as a correctly spelled word. You will still have to edit your material even though your spelling is corrected by SPELL WIZARD. Since this is a program you can use without a word processor you can correct any mistakes in any files created with Atari 2.0S DOS. This also means any files or word processor (LETTER PERFECT) which did not use Atari 2.0S DOS can't be used with SPELL WIZARD.

SPELL WIZARD is a menu driven program, just select the option you need and in most instances you will be answered with a prompt YES or NO. Then while it is reading your document it records the total number of words it has read, and how many of the words were new to it. This information is displayed in labeled windows. All the words it finds which were either misspelled or were not in the dictionary will be displayed for you. It will then ask if you want the word corrected. If you answer YES it will then do so. The words not appearing in the dictionary you will have to check for spelling. If you find they are correct then it will ask if you want them entered in the dictionary.

Another nice feature is you may exit SPELL WIZARD and automatically boot to your word processor after you have finished with the program without shutting down and starting up the system again.

There are a great many programs on the market which are by themselves very good, but the documentation coming with them is terrible. Many a good program has been left by the wayside because the manufacturer provided instructions which didn't fully explain how the program works, or made the program so complicated one didn't want to use it. Thanks to Datasoft for making the documentation for SPELL WIZARD complete with a minimum of instruction.

Once you start using this program you will never want to be without it again.

—Larry Gold

STONE OF SISYPHUS

(Chameleon Software, under \$40.00)

If you like D&D games then this is a fair one for you. There is treasure, monsters, and magic galore as you travel through the mazes and dungeons. The object is to collect as much treasure as you can and get out to spend it, while using your wits to do so.

You do have to equip a character with suitable weapons, armor, and various skills, and then send him (her) on his (her) way.

The graphics in this game are all seen from the point of the person heading deeper into the dungeon and at the same time the player has a choice of escaping from the situation or carrying on.

The program is written in Basic and obviously needs at least one player. About a month is needed to complete the game, and the age group is 14 to adult. There is no sound, but in this type of game none is needed. There are two disks, but only one disk drive is needed to play. There is also a game saving feature.

On a scale of 1 to 10 I rate this game a seven. There really isn't anything to make this game stand out from other D&D games.

I think the one saving grace of these text-graphic adventure games is they do tend to teach the player a certain type of logic, which I'm sure will one day come in handy to help one think through one's problems.

Unfortunately after one plays these type of games for any length of time they all begin to look the same, then the player loses interest.

STONE OF SISYPHUS should hold the interest of a new D&D player for sometime and I recommend it for this type of game player.

(The above spelling was corrected by SPELL WIZARD and any misspelled words are the courtesy of the editors.) Boo! —JB, ed.

—LARRY GOLD



BUMPAS REVIEWS

I want to thank all you folks who wrote to explain to me how Filemanager 800+ handles the "9 pages of 20 records each". Turns out all I had to do is read the documentation where it says there can be a maximum of 20 fields per record. Everyone pointed this out to me, as it was the error I made. I tend to assume things about software when I think I know how something works. I just assumed there must be more than 20 fields available for each record. I was so convinced, I didn't even try to find out otherwise in the documentation. I merely tried to find out how to get to a second page after filling the first up with fields! Thanks, again. The lesson I learn from this is to read good documentation when I have it. We're lucky Synapse writes good documentation.

WALLWAR

SierraVision, Sierra-on-Line Bldg., Coarsegold, CA 93614, has produced an innovative game on disk for the Atari requiring 40k and joysticks.

The game is not your usual "shoot 'em up", in spite of the fact the game involves almost constant shooting. The graphics are done in very fine resolution, and the use of color is very pleasing. The sounds are appropriate.

The first innovation one finds is a demonstration game which gives the feel of the game. When you jump in, you may play the Atari, or another human opponent.

The screen contains 2 "microbots", each trying to protect a "plasma field", separated by a 5-layer light barrier which must be penetrated.

The next innovation is the indirect nature of the violence inflicted upon one's opponent. A player cannot "kill" the opposing microbot by shooting him. If you hit him, you can slow him down and temporarily damage him. That is all. The way you kill him is by destroying his plasma field, which he can protect by throwing his body in the way of your shots, or by using "laser" beams to destroy the incoming missiles.

—Jim Bumpas

FORTH ON THE ATARI

—Learning by Using
by E. Floegel

This book is only 118 pages long which in itself will tell you the author uses a "once over lightly" approach to Forth programming. It is not as comprehensive as Brodie's **STARTING FORTH** but it does have the advantage of having been written exclusively for the Atari computer.

While **Starting Forth** remains my personal "Forth Bible", this book does do a good job using examples to show specific Forth programming complexities which, to the uninitiated, can otherwise seem very conceptual and abstract.

The sample programs in Chapter 5 deal specifically with graphics and sound — areas not covered in much depth in other Forth books written to date. These samples are invaluable to me just to see how someone else solves some of the same Forth programming dilemmas I have faced.

Chapter 6, covering test and string handling, helps clear away some of the abstraction and mystique I find in Brodie's approach, which to me occasionally assumes too much prior programming knowledge in certain areas.

This book is not a text allowing anyone unfamiliar with Forth to learn it all. It is, however, a very good supplementary text which helps smooth out some of the rough spots. A plus to the approach of this book is its attempt to teach by example rather than precept alone. This is helpful to anyone struggling with Forth programming concepts.

My main gripe about this publication is either (1) the author is a lousy speller or (2) the typesetter was smashed or (3) the editor was "inexperienced". I have never read a book with as many spelling and/or typographical errors. It adds humor at times, but after a while it's annoying. I suspect the publication was a rush job, no doubt to appease Atari owners always on the watch for new materials.

Forth programmers will get enough from the book to warrant purchasing it. Beginners will not find the book comprehensive enough by itself, but in combination with other Forth texts it is a good investment.

—Graham Smith, ACE Eugene

(by ELCOMP, 53 Redrock Lane
Pomona, CA 91766 \$7.95)

APRIL MEETING

4/13/83 7:30 PM

LCC FORUM 308

SPECIAL MEETING

BRING YOUR FREINDS

TYPESETTING FROM YOUR COMPUTER

ATARI OWNERS: If you have a modem, text editor, and communications program to send ASCII files, you should consider the improved readability and cost savings provided by **TYPESETTING** your program documentation, manuscript, newsletter, or other lengthy text instead of just reproducing it from line printer or daisy-wheel output. Computer typesetting by telephone offers you high quality, space-saving copy that creates the professional image you want! Hundreds of type styles to choose from with 8 styles and 12 sizes "on line." And it's easy to encode your copy with the few typesetting commands you need.

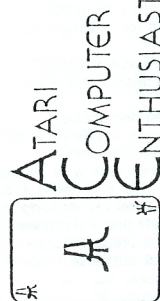
COMPLETE CONFIDENTIALITY GUARANTEED

— Bonded for your protection —

PUBLICATION DESIGN, EDITING, & PRODUCTION

Editing & Design Services
Inc.

30 East 13th Avenue Eugene, Oregon 97401
Phone 503/683-2657



3662 Vine Maple Dr. Eugene OR 97405

FIRST CLASS MAIL

Atari Computer Enthusiasts

A.C.E. is an independent computer club and user's group with no connection to the Atari Company, a division of Warner Communication Company. We are a group interested in educating our members in the use of the Atari Computer and in giving the latest News, Reviews and Rumors.

All our articles, reviews and programs come from you, our members.

Our membership is world-wide; membership fees include the A.C.E. Newsletter. Dues are \$10 a year for U.S., and \$20 a year Overseas Airmail and include about 10 issues a year of the ACE Newsletter.

Subscription Dep't: 3662 Vine Maple Dr., Eugene, OR 97405
President—Kirt Stockwell, 1810 Harris #139, Eugene, Or 97403 / 503-683-3005
Vice Pres—Larry Gold, 1927 McLean Blvd., Eugene, Or 97405 / 503-686-1490
Secretary—Charles Andrews, POB 1613, Eugene, Or 974401613 / 503-747-9892
Librarian—Chuck and Jody Ross, 2222 Ironwood, Eugene, OR 97401 / 503-343-5545
Editors—Mike Dunn, 3662 Vine Maple Dr., Eugene, Or 97405 / 503-344-6193
Jim Bumpas, 4405 Dillard Rd., Eugene, Or 97405 / 503-484-9925
E.R.A.C.E. (Educational SIG Editor)—Ali Erickson, 295 Foxtail Dr., Eugene, Or 97405 / 503-687-1133

Send a business-size SASE to the Ross' for the new, updated ACE Library List!!

Bulletin Board
(503) 343-4352

On line 24 hours a day, except for servicing and updating. Consists of a Tare equipped 48K Atari 400, 2 double-density disk drives, an Atari 825 printer, a Hayes SmartModem; running the ARMUDIC Bulletin Board software written by Frank L. Huband. See the Nov '82 issue for complete details.